Universität Zürich[UZH]

Department of Informatics

Agile, Vision-Based Quadrotor Flight: from Active, Low-Latency Perception to Adaptive Morphology

Dissertation submitted to the Faculty of Business,
Economics and Informatics
of the University of Zurich

to obtain the degree of
Doktor der Wissenschaften, Dr. sc.
(corresponds to Doctor of Science, PhD)

presented by
Davide Falanga
from Torre del Greco, NA, Italy

approved in February 2020

at the request of
Prof. Dr. Davide Scaramuzza
Prof. Dr. Roland Siegwart
Prof. Dr. Sami Haddadin
Prof. Dr. Nathan Michael

Universität Zürich[UZH]

Department of Informatics

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, February 12, 2020

The Chairman of the Doctoral Board: Prof. Dr. Thomas Fritz

To my family.

Acknowledgements

I would like to thank my advisor Prof. Davide Scaramuzza for selecting me as a PhD student and giving me the opportunity and freedom to pursue many interesting ideas and projects. Also, I would like to thank Tammy Tolcachier for her crucial, unvaluable and continuous help with both work-related and private issues I had in these years, during which she has been a fundamental reference point for my life in Switzerland.
This thesis is the result of many collaborations and fruitful discussions. I therefore want to express my gratitude to Elias Mueggler, Matthias Faessler, Alessio Zanchettin, Kevin Kleber, Philipp Foehn, Alessandro Simovic, Suseong Kim, Jeff Delmerico, Titus Cieslewski, Julien Kohler, Naveen Kuppuswamy, Mac Schwager, Riccardo Spica, Eric Cristofalo, Eduardo Montijano, Antonio Franchi, Dario Floreano, Stefano Mintchev, Bianca Sangiovanni and Peng Lu for the active collaborations we had.
RPG has been a second family to me, and I had the lack of enjoying several funny moments thanks to the contribution of the fantastic people I met: Giovanni Cioffi, Roberto Tazzari, Guillermo Gallego, Henri Rebecq, Zichao Zhang, Antonio Loquercio, Elia Kaufmann, Mathias Gehrig, Thomas Laengle, Kunal Shrivastava, Manuel Sutter, Yunlong Song, Manasi Muglikar, Javier Hidalgo Carrio, Dimche Kostadinov, Christian Pfeiffer, Dario Brescianini, Junjie Zhang, Matia Pizzoli, Manuel Werlberger, Francisco Javier Perez Grau, Toni Rosinol Vidal, Raphael Meyer, Michael Gassner, Flavio Fontana, Christian Forster and Reza Sabsevari.
I also had the pleasure to work with great students, namely Amedeo Fabris, Barza Nisar, Harshit Khurana, Anna Leidi, Nils Funk, Robin Scherrer, Maria Chiara Giorgetti, Valentin Wuest and Kevin Egger.
I would like to thank Prof. Roland Siegwart, Prof. Nathan Michael and Prof. Sammi Haddadin for reviewing my thesis and their valuable feedback.
Last, but not least, I thank my family for their help during these intense five years. In particular, I want to thank my wife, Roberta, for being always supportive and tolerant with me, for motivating me to pursue my goals, and for standing by me in the hardest moment of this journey. Without you, I would not be the man I am today, and I will always be thankful for this. A huge thanks goes to my parents, to whom I owe everything I have today, and who have been always very close to me, despite the large distance keeping us apart. Finally, if I reached this target it is certainly also thanks to my brother, and best friend, Gianluca, who always knows how to cheer me up.

Zurich, December 2019 D. F.

Abstract

Aerial vehicles are leading the robotics revolution, generating new industries with a potential market value of several billion dollars. In the future, flying robots will deliver goods directly to our homes, inspect large structures that are unsafe for or inaccessible by human operators, surveil our cities to guarantee safety, transport people over short distances, and perform search-and-rescue missions to react to natural disasters promptly. Therefore, aerial robotics is destined to be responsible for a significant paradigm shift in our everyday life, with quadrotors playing a crucial role thanks to their agility, simple electromechanical design, and vertical take-off capabilities. However, for this to happen, several important steps forward in the direction of increased autonomy, maneuverability, and robustness must be taken.

For quadrotors to be completely autonomous, it is necessary for them to only rely on onboard sensors and computers, and to be able to access complex areas. This poses severe challenges when it comes to deploying these systems in real-world scenarios, which are typically not designed to favor robot perception and navigation. Lack of visual texture, fast-moving obstacles, and buildings not accessible by fixed-sized quadrotors are only some examples of the unsolved challenges in terms of sensing, planning, and control that need to be tackled to increase the level of autonomy of flying robots. Allowing a quadrotor to move in a way that favors perception, equipping it with sensors and algorithms for low-latency obstacle detection and avoidance, and enabling shape-shifting capabilities bring the benefit of unlocking the full potential of these vehicles.

This thesis focuses on motion planning and control methods that enable vision-based quadrotors to: (i) plan and execute trajectories that improve visual perception; (ii) sense and avoid fast obstacles by leveraging event cameras; (iii) change their shape and size while flying. This thesis also presents contributions in the application of quadrotors, such as a system for autonomous, vision-based landing on a moving platform. The following is a list of contributions:

- The first quadrotor that can actively change its shape while flying, guaranteeing stable flight at all times, independently of the morphology. This vehicle was the recipient of the first prize for the category Aerospace and Defense at the 2019 NASA Tech Briefs contest and won the *Most Innovative Drone* award at the 2019 Drone Hero contest.

- The first method that allows a quadrotor to fly through narrow inclined gaps in

an agile maneuver based only on onboard sensing and computation.

- The first mathematical analysis of the impact of perception latency on the maximum speed a robot can achieve in a *sense-and-avoid* task.
- A method to detect fast-moving obstacles with low latency using an event camera, and to avoid them using a reactive approach.
- A perception-aware Model Predictive Control scheme for quadrotors, capable of trading-off perception and action objectives in order to execute a given task while keeping visible some points of interest.
- A framework to let a quadrotor autonomously land on a moving platform using only onboard sensors and computation.

List of Contributions

Google Scholar Profile: http://bit.ly/davide-falanga-scholar

Journal Publications

- **Davide Falanga**, Kevin Kleber, and Davide Scaramuzza. "Low Latency Avoidance of Dynamic Obstacles for Quadrotors with Event Cameras". Under review in: *AAAS Science Robotics*.Links: Video

- **Davide Falanga**, Suseong Kim, and Davide Scaramuzza. "How Fast is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid". In: *IEEE Robotics and Automation Letters (RA-L)* 4.2 (2019), pp. 1884–1891. DOI: 10.1109/LRA.2019.2898117
Links: PDF, Video

- **Davide Falanga**, Kevin Kleber, Stefano Mintchev, Dario Floreano, and Davide Scaramuzza, "The Foldable Drone: A Morphing Quadrotor that can Squeeze and Fly". In: *IEEE Robotics and Automation Letters (RA-L)* 4.2 (2019), pp. 209–216. DOI: 10.1109/LRA.2018.2885575
Links: PDF, Video

- Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, **Davide Falanga**, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, Guido de Croon, Sunyou Hwang, Sunggoo Jung, Hyunchul Shim, Haeryang Kim, Minhyuk Park, Tsz-Chiu Au, and Si Jung Kim. "Challenges and implemented technologies used in autonomous drone racing". In: *Springer: Intelligent Service Robotics Series* 12.2 (2019), pp. 137–148. DOI: 10.1007/s11370-018-00271-6
Links: PDF

- Barza Nisar, Philipp Foehn, **Davide Falanga**, and Davide Scaramuzza. "VIMO: Simultaneous Visual Inertial Model-based Odometry and Force Estimation". In: *IEEE Robotics and Automation Letters (RA-L)* 4.3 (2019), pp. 2785–2792. DOI: 10.1109/LRA.2019.2918689
Links: PDF

- Suseong Kim, **Davide Falanga**, Davide Scaramuzza. "Computing The Forward Reachable Set for a Multirotor Under First-Order Aerodynamic Effects". In: *IEEE Robotics and Automation Letters (RA-L)* 3.4 (2018), 2934–2941. DOI: 10.1109/LRA.2018.2848302
Links: PDF

- Matthias Faessler, **Davide Falanga**, and Davide Scaramuzza. "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight". In: *IEEE Robotics and Automation Letters (RA-L)* 2.2 (2017), pp. 476–482. DOI: 10.1109/LRA.2016.2640362
Links: PDF, Video

Peer-Reviewed Conference Papers

- R. Spica, **Davide Falanga**, Eric Cristofalo, Eduardo Montijano, Davide Scaramuzza, and Mac Schwager. "A Game Theoretic Approach to Autonomous Two-Player Drone Racing". In: *Robotics: Science and Systems (RSS)* 2018. DOI: 10.15607/RSS.2018.XIV.040
  Links: PDF, Video

- **Davide Falanga**, Philipp Foehn, Peng Lu, and Davide Scaramuzza. "PAMPC: Perception-Aware Model Predictive Control for Quadrotors". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* 2018. DOI: 10.1109/IROS.2018.8593739
  Links: PDF, Video, Software

- **Davide Falanga**, Elias Mueggler, Matthias Faessler, and Davide Scaramuzza. "Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing". In: *IEEE International Conference on Robotics and Automation (ICRA)* 2017. DOI: 10.1109/ICRA.2017.7989679
  Links: PDF, Video

- Philipp Foehn, **Davide Falanga**, Naveen Kuppuswamy, Russ Tedrake, and Davide Scaramuzza. "Fast Trajectory Optimization for Agile Quadrotor Maneuvers with a Cable-Suspended Payload". In: *Robotics: Science and Systems (RSS)* 2017. DOI: 10.15607/RSS.2017.XIII.030
  Links: PDF, Video

- **Davide Falanga**, Alessio Zanchettin, Alessandro Simovic, Jeffrey Delmerico, and Davide Scaramuzza. "Vision-based Autonomous Quadrotor Landing on a Moving Platform". In: *IEEE/RSJ International Symposium. on Safety, Security and Rescue Robotics (SSRR)* 2017. DOI: 10.1109/SSRR.2017.8088164
  Links: PDF, Video

Awards

- Winner of the 2019 NASA Tech Briefs *Create the Future* contest for the category "Aerospace & Defense" (Official Page)
- Winner of the Drone Hero Award, "Most Innovative Drone" category, 2019 (Official Page)
- RSS Best Paper Student Paper Award Finalist, 2017
- Winner of the IROS 2018 Autonomous Drone Racing Competition (Official Page)

Open-Source Software

- Perception-Aware Model Predictive Control for Quadrotors (PAMPC).
  - http://github.com/uzh-rpg/rpg_mpc

Contents

1 Introduction

This thesis presents algorithms for motion planning and control of autonomous, vision-based quadrotors. It focuses on two main topics: (i) bridging the gap between perception and action by addressing some challenges deriving from vision-based sensing, such as the need for visual texture, degradation of performance at high speed due to motion blur, and latency; (ii) extending the maneuverability of quadrotors, proposing a novel morphing quadrotor design that goes beyond the rigid mechanical structure proposed in the literature to allow shape and size adaptation in flight.

This work is split into three parts. First, it addresses the problem of coupling perception and action using active vision to manipulate the viewpoint of a camera mounted onboard a quadrotor, to execute a given task (e.g., following a reference trajectory) while simultaneously obtaining as much visual information from the surrounding environment as possible. Second, it focuses on perception latency, analyzing the impact it has on the agility of an autonomous quadrotor, and presenting a framework to detect and reactively avoid fast, dynamic obstacles using event cameras. Finally, it presents the foldable drone, a quadrotor equipped with four additional servo-motors that allow each arm to rotate around the main body of the vehicle. By doing so, the system is capable of adapting its morphology to the task at hand, allowing, for example, to pass through spaces that are narrower than the robot size.

This thesis is structured in the form of a collection of papers. Self-contained publications follow an introductory section that highlights the concepts and ideas behind the thesis in the appendix. The next sections discuss the working principle, history, current, and future applications, advantages, and challenges of quadrotors. Section 1.2 provides a brief overview of the history of hovering rotorcrafts, bridging the gap between the first prototypes and the currently available systems, and highlights their current and future applications. Section 1.4 summarizes the state of the art in autonomous, vision-based quadrotor flight, with a focus on the three topics covered by this thesis. Section 1.3 motivates and states the research objectives of this dissertation. The papers in the appendix are summarized in Chapter 2. Finally, Chapter 3 provides future research directions.

1.1 The Robotic Revolution: Current Status and Challenges

We live on the edge of a technological revolution with a potentially disruptive impact on our everyday life. Our society is currently witnessing tremendous, fast-pacing progress in the development of intelligent machines that can animate and interact with the physical world. Artificial Intelligence (AI) and Robotics represent the core of what has been defined as the *fourth industrial revolution*[1] which, differently from the first three, will allow for the first time machines to think and move in the real world, all by themselves. In a not so distant future, intelligent cyber-physical systems will significantly change our society, our cities, our life. Robots will become ubiquitous, following the path outlined by personal computers at the end of the 20th century and giving birth to the new term, and consequently to the new market of *personal robots*. Owning a private robot will become as common as owning a laptop, a smartphone or a smartwatch, compared to which robots have a substantial advantage that will make the robotic revolution significantly more impactful than the previous ones: they can execute physical actions.

The so-called *Industry 4.0* gives clear evidence of the potential impact of robotics technologies. Since AI entered the scene in the last decade, industrial manufacturing went through a sequence of massive changes that improved the production of any sort of goods in terms of effectiveness, reliability, cost, and safety, with clear benefits for both companies and consumers. At the same time, however, industrial applications of robotics shade light on the inappropriateness of current state-of-the-art technologies outside the industrial context. Current robots are smarter, faster, and safer than their counterparts from 30 or 40 years ago. Nevertheless, the vast majority of them still operate within clearly defined boundaries, often behind safety fancies, and repeatedly execute a single well-structured task in an extremely controlled environment. This is true not only for industrial robots since these limitations often affect other robotic systems. If one looks at the use of robots outside manufacturing plants, there are very few examples of robotic systems in our everyday life. Autonomous mobile robots recently became pretty standard in large storage areas to automatize transportation of goods inside warehouses. Autonomous vacuum cleaners represent the main domestic application of robots. These robots can operate only in specific environments with a clear structure, and often exploit very simple navigation algorithms. Additionally, they need some coordination across multiple agents or assume some geometric properties of the environment they navigate through. Finally, they sometimes require some human intervention to operate. Therefore, they are not suited for deployment in more complex scenarios.

The main bottleneck to the diffusion on a large scale of robotic technologies is the lack of algorithms that allow them to deal with real-world environments. These are typically

[1] http://www.foreignaffairs.com/articles/2015-12-12/fourth-industrial-revolution/

not designed to favor robotic systems, and often pose severe challenges for perception, control, motion planning, and decision making. For example, visual features necessary for onboard perception are usually not spread across real-world environments in a way the favors current algorithms used for localization and mapping, or they might not suffice in number to provide enough information for robust sensing. Additionally, dynamic scenarios are currently hard to handle for current robots, since they require fast perception and planning at high rates if one wants to guarantee safety for all the agents, both human and robots, sharing the same space. Finally, structures built for humans might not be suitable for robotic navigation due to the shape and size of the platform.

The previous examples show that to be capable of executing tasks in a large variety of real scenarios, robots must be able, among several other things, to: (i) gather as much information from their onboard sensors by coupling perception and action to move in a way that favors robust localization and sensing; (ii) be equipped with fast perception-to-action loops, to promptly react to sudden changes in the environment, as moving obstacle; (iii) adapt their mechanical properties, such as their shape and size, to navigate through complex areas, where adaptive morphologies can provide added value to reach otherwise unreachable areas and increase the robot's maneuverability. The goal of this thesis is to address these open challenges to develop novel algorithms that can extend the variety of scenarios and tasks that autonomous quadrotors can handle.

1.2 Past, Present, and Future: History and Applications of Flying Robots

In recent years, more and more robotic systems are leaving the boundaries of research laboratories and manufacturing sites to be deployed in new scenarios, interacting with humans and navigating through complex environments. If robots are becoming increasingly popular, much of the merit goes to aerial robots, and especially to small-scale multirotors. In this regard, quadrotors are beyond any doubt the most common form of flying robots, as confirmed by the fact that they can be easily found in almost any consumer electronics store. The level of autonomy of these vehicles is still pretty limited compared to their potential. Mass-scale deployment of autonomous quadrotors is still far in time since, apart from few isolated cases, they are often manually piloted or somehow remotely assisted by a human operator. Nevertheless, quadrotors have significantly contributed to bridging the gap between the research community and the masses, for whom robots mostly belonged to Sci-Fi movies before the advent of flying robots in their everyday life.

In the next subsections, I will provide an overview of the past, the present, and the future of these vehicles. I will start with the first concepts of manned multirotors,

which inspired the development in the past decades of small-scale rotorcrafts. I will then move on to the recent progress in the field of unmanned Micro-Aerial Vehicles (MAVs), which unlocked the doors of entirely new markets and are revolutionizing the robotics and aerial industry. Finally, I will highlight some of the main challenges to be solved to enable these robots to autonomously navigate in real-world scenarios, stating how this thesis contributes to them.

1.2.1 History of Multirotors: from the First Concepts to Small-Scale, Unmanned Vehicles

Quadrotors have become a popular research platform during the past two decades; however, the first ideas of flying machines equipped with multiple rotors date to the beginning of the 20th century [2]. Before describing the current state of the art in this field and looking ahead to highlight the future steps necessary for it to progress, it is worth to spend a few words about how everything started and to provide some historical details about when the idea of flying vehicles capable of hovering thanks to the use of rotors equipped with propellers was conceived.

The first traces of these kinds of flying machines date back to the 15th century: the *Vite Aerea* of Leonardo Da Vinci is probably the oldest documented draft of a rotorcraft, though it never became more than a simple conceptual idea accompanied by some schematic drawings. In the next few centuries, several attempts of realizing somewhat similar concepts to what Da Vinci proposed where realized, for example, Mikhail Lomonosov's Aerodynamic, the Helicopter Toy by Launoy and Bienvenu, and the so-called *governable parachute* by Sir George Cayley. However, there exists no proof or documentation guaranteeing that any of those machines successfully took off and flew as expected by their respective creators.

To see the first successfully hovering vehicles, it is necessary to wait until the beginning of the last century. Two fundamental milestones in the field of hovering vehicles were both placed in 1907, when Bréguet-Richet, author of the Gyroplane, and Paul Cornu, creator of the first system capable of hovering, presented to the world their revolutionary, yet premature for that time, ideas. In 1920, Étienne Oehmichen realized a rotorcraft design equipped with four rotors, each with eight blades, with only one engine responsible for actuation. Two years later, in 1922, Jerome-de Bothezat presented the Flying Octopus, a six-bladed multirotor with an X-shaped structure currently considered the first successful ancestor of today's multicopters, and laid the foundation for current designs. These pioneeristic attempts of realizing aerial vehicle capable of hovering were conceived one century before the robotics community tried to achieve autonomous unmanned flight, at the turn of the 90s and 2000s. In this sense, the

[2] http://www.wired.com/2007/12/gallery-helicopter/

1991 International Aerial Robotics Competition [3] is among the oldest attempts from researchers and practitioners to develop autonomous flying robots. Fast forward to our days, the DARPA Fast Lightweight Program (2015-2018) promoted the development of fast, autonomous quadrotors capable of navigating all by themselves at high speed. In between, almost 20 years of research and efforts from the community brought these vehicles from the status of theoretical concepts to be the most common aerial platform in the robotics field.

The idea of building a small-scale multirotor with four propellers connected through a rigid-body housing all the necessary equipment for unmanned flight was conceived towards the end of the 19th century. Commonly known as *X4 Flyers*, quadrotors were already available on the market in the second half of the 90s, with products like the and Roswell flyer (from Area Fifty-One Technologies, 1996), the DraganFlyer (from RCToys), and the Keyence's Engager and Gyrosaucer. However, it is necessary to wait until the beginning of this century for quadrotors to attract the interest of the robotics research community [73, 74, 147]. These vehicles were significantly larger (several tens of centimeters from tip to tip) and heavier (typically a few kilograms) than current micro- and nano-scale quadrotors. It is with the development of electronic Inertial Measurement Units (IMU), thanks to the progress in the field of consumer electronics (i.e., gaming devices and smartphones), that it was possible to equip quadrotors with small, lightweight, cheap devices capable of providing inertial data for attitude stabilization, allowing multirotor aircrafts to become significantly smaller and more practical. The availability of affordable motion-capture systems first, and then the algorithmic progress in the area of Visual-Inertial Odometry (VIO), recently made it possible to obtain a full state estimate in GPS-denied environments. Combining this state estimate with novel control algorithms for position stabilization capable of running on small-scale onboard computers, allowed the execution of the first fully autonomous flights in the past decade.

1.2.2 Applications of Quadrotors

Among the different types of Micro Aerial Vehicles, quadrotors play a significant role thanks to their simple mechanical structure and their safety. Indeed, they are composed of a frame equipped with four rotors, each of them mounting a propeller for actuation. The motion of the vehicle is obtained by suitably tuning the rotational speed of each propeller to generate the forces and torques necessary to let the vehicle move in space. The main body can host sensors, computation units, and other payloads to be carried. The main advantage of quadrotors against other aerials vehicles, such as fixed-wing aircrafts, is their versatility: these robots are capable of locking to a hover position, to provide an aerial view of the environment, as well as flying at high-speed for quick exploration. Furthermore, they can take-off and land vertically, rendering

[3] http://www.aerialroboticscompetition.org/mission1.php

field deployment significantly more straightforward.

Thanks to these peculiarities, quadrotors are nowadays the most popular form of flying robots, both in the research community and in the consumer market. In this sense, in the last few years, quadrotors had a disruptive impact on the industry, with the creation of entirely new markets worth several billion of US Dollars, and the birth of novel applications with tremendous potential. Some of these applications are already a reality, while others will become soon. In the following, I will list the major areas where quadrotors did or will, in the future, become extremely popular and provide added value.

Aerial imaging. Aerial photography and videography currently represent the largest market in terms of monetary value. Aerial imaging for both private (i.e., shooting videos and pictures from an aerial perspective for personal use) and public or enterprise applications (i.e., mapping and monitoring for real estate, agriculture, civil engineering and inspection, defense, law enforcement, counter-terrorism) was valued around 1.4 billion USD in 2017, with a projected growth to more than 4 billion USD in 2025 [4]. The use of quadrotors as flying cameras represents, and probably will represent in the next years, the main application of these vehicles outside research.

Aerial delivery. Autonomous last-mile delivery represents the main threat to the role of aerial imaging as the dominator in the field of aerial robotics. MAVs, including quadrotors and hybrid vehicles as tail sitters, represent a possible solution to the problem of promptly delivering goods within short ranges. Boosted by the efforts from numerous companies interested in developing novel technologies for aerial transportation, either to deliver commercial products (as for example aimed by, among the others, Amazon Prime Air, Google Wing, Uber AIR and Walmart) or life-critical goods (such as blood samples and medicines, as intended by Matternet and Zipline), the market of aerial transportation using drones has a projected value of more than 90 billion USD [5] by 2030. Several challenges still need to be addressed in this area, both in terms of regulation and technological problems to be solved, but the day when drone delivery becomes truth does not seem to be too far in time.

Aerial mobility. The field of personal transportation is going to soon go through a disruptive revolution that will change the way we conceive mobility. If, on the one hand, driverless technologies will soon allow cars to drive autonomously on our roads, recently there have been several companies (Uber, Airbus, Volocopter, to mention a few) that started realizing prototypes of flying machines for personal transportation. Their

[4] http://bit.do/aerial-imaging-market
[5] http://bit.do/aerial-delivery-market

size is significantly larger than MAVs, but several of the technological and engineering challenges to allow autonomous quadrotor flight also belong to this new category of transportation vehicles. The forecast for this market is also impressive, with a projected value of 3.1 billion USD by 2023, which is expected to grow up to almost 8 billion USD in 2030 [6]

Search and rescue. Search and rescue is another domain that could greatly benefit from aerial robots capable of autonomously exploring unstructured and potentially dangerous environments [34]. One day, autonomous flying robots will play a significant role in search-and-rescue missions, where a fast response is crucial. They can provide a *birds-eye* view of a scene and, if necessary, a real-time 3D reconstruction of the area of interest, helping rescuers to make critical decisions when it comes to deciding how to act to help victims promptly [43]. Applications of robotic technologies to this field are inspection in post-disaster scenarios, localization of survivors, and rescue of missing people. Additionally, MAVs can navigate through complex environments, some of which might not be accessible by the rescuers. For example, quadrotors can enter and exit (semi-)collapsed buildings through narrow gaps in the case the usual means of access to them are not available [46]. This capability can significantly speed-up the execution of time-critical rescue missions, rendering them safer for the rescuers and more effective.

E-Sports. Thanks to their impressive agility and acrobatic capabilities, quadrotors recently gave birth to a completely new sport, known as *drone racing*. Several enthusiasts from all over the world challenge each other regularly during local and international competitions, and an official Drone Racing League [7] was recently created to gather them under a common umbrella. The market value of drone racing is currently around 3 billion USD [8], with forecast growth of around 20% in the next few years. These races typically require a skilled human pilot to complete a track in as little time as possible, but both the research community and industry have recently started investigating technologies to allow quadrotors to race in a completely autonomous fashion. An Autonomous Drone Race already takes place regularly every year during the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) since 2016, and recently the Drone Racing League, together with Lockheed Martin, announced the first autonomous drone racing on a large scale [9], with a 1 million USD cash prize. This is just the beginning of a new, revolutionary *e-sport*, which has the potential to gather as much attention from the public and the media as car and motor racing.

[6] http://bit.do/air-mobility-market
[7] http://thedroneracingleague.com
[8] http://bit.do/drone-racing-market
[9] http://bit.do/alphapilot

1.3 Research Objectives

Looking ten years back, it is easy to notice the impressive progress made in the field of autonomous quadrotors. This was possible thanks to both the availability of novel, cheaper and more effective off-the-shelf components (such as lighter sensors, more powerful onboard computers, more efficient mechanical parts), and the algorithmic innovations in the areas of vision-based perception (e.g., Visual-Inertial Odometry, mapping), motion planning, control and decision making. Nevertheless, if today we do not see autonomous quadrotors executing all the tasks previously mentioned tasks, the reason is a large number of open challenges that need to be addressed.

The amount and variety of these challenges are so deep that it is certainly not possible to tackle all of them within a single Ph.D. thesis since the joint effort of the entire robotics community is necessary. However, during my doctoral work, I identified some key challenges in this field, listed below, in order to provide solutions to them. The objectives of this thesis are twofold: (i) allowing quadrotors to fly autonomously using onboard, vision-based perception; (ii) extending the variety of areas that they can navigate through by adapting the robot's morphology to the task at hand and the properties of the environment.

Tight Coupling of Perception and Action. The goal of the first objective is to consider the main limitations of vision-based perception within motion planning and control loops. Perception algorithms based on cameras allow a robot to perceive its surroundings, but require enough visual information (e.g., texture), their performance can degrade due to motion blur, and introduce latency. In this work, I tackle these issues by developing algorithms that tightly couple perception and action, and leveraging low-latency event cameras to propose a framework for the avoidance of fast-moving obstacles.

State-of-the-art in quadrotors flight often relies on external infrastructures, such as motion-capture systems, to localize the robot. These systems are typically expensive, require a non-negligible time for installation and calibration, and strongly limit the potential of aerial vehicles by reducing the flyable space to the region where they provide coverage. Therefore, they are not suitable for real-world scenarios. As of today, onboard vision represents the only viable solution for self-localization. Providing a robot with cameras for localization represents a major step forward in autonomous flight. However, off-board state estimation, for example, through motion-capture systems, has the following advantages when compared to vision-based state estimation: (i) the state estimate is *available at all times*, with no interruption; (ii) the state estimate is very accurate (i.e., *sub-millimeter*) and with *constant noise covariance* within the tracking volume, (iii) the state is estimated with very *low latency* (less than 10 ms) and high frame rate (more than 100 Hz). By contrast, onboard vision is more challenging: (i) the

state estimate can be intermittent (i.e., tracking may be lost); (ii) the uncertainty of the state estimate increases with the distance from the scene and is also strongly affected by the type of structure and texture of the scene, as well as by the motion of the camera with respect to the environment; (iii) it has higher latency due to the time to capture, transfer and process frames (50–100 ms, depending on the complexity of the perception task), and the frame rate is typically less than 100 Hz. For these reasons, to get the best out of a vision-based perception algorithm, one cannot treat perception and control separately, but rather it is necessary to tightly couple them.

One of the goals of this work is to develop motion planning and control algorithms that simultaneously consider action and perception objectives. This problem is particularly challenging for underactuated systems such as quadrotors, where the kind of motion the vehicle can perform must satisfy the system dynamics. To do so, this thesis investigates optimization-based motion planning techniques in order to jointly consider action objectives (i.e., the need to execute a given task by following a reference trajectory driving the vehicle towards its goal) and perception objectives (i.e., the necessity to perform such task while providing perception algorithms with sufficient visual information), satisfying at the same time the system dynamics and the actuation limitations of the platform. Both closed-form and numerical optimization techniques are exploited, and the effectiveness of the proposed methods are validated with real-world experiments in complex tasks such as traversing narrow gaps, flying at high-speed while keeping some points of interest always visible, and flying in an area with very poor visual information.

Low-Latency Sensing and Decision-Making. Latency represents another limitation of vision-based perception. Perception latency can pose severe bounds to the agility of a robotic platform: the smaller the latency, the faster a robot can move. Generally, perception latency is due to two factors: the time required to obtain a measurement, and the time necessary to process it in order to extract valuable information. The majority of the works based on onboard perception relies on frame-based cameras, which introduce latency due to the exposure time and the transfer time. Additionally, processing an image can be computationally expensive, especially with high-resolution cameras.

I dedicated a part of my work to studying the use of novel, low-latency event-cameras, which do not produce frames but rather events, defined as the response to changes of intensity at each pixel location. The advantages of this new sensor are multiple, including a lower amount of information that is transferred and processed, and significantly lower latency. In this dissertation, I investigate the benefits of adopting event-cameras against standard cameras for a *sense-and-avoid* task. To do so, I derived a mathematical relation between the maximum velocity a robot can safely navigate through an unknown environment, the parameters of its perception system (latency

and sensing range) and its mechanical properties (the maximum acceleration it can produce). As a case study, I analyzed and compared frame-based and event-based sensing. Furthermore, I propose a low-latency, event-based reactive scheme to avoid fast-moving objects complementing slower, but more accurate, navigation systems, in order to provide quadrotors with an effective device to guarantee safety when flying in dynamic environments.

Adaptive Morphology for Enhanced Maneuverability. Apart from investigating techniques to deal with the main limitations of onboard perception, this work also aims at proposing a morphing quadrotor platform that goes beyond the standard rigid mechanical structure that characterized these vehicles in the last decade. The proposed *foldable drone* can adapt its morphology by rearranging the position of the four arms around the main body, allowing it to change its shape and size. By doing so, it can enter spaces that would not be accessible due to the silhouette of the robot. This capability turns out to be particularly effective in search-and-rescue missions, where it might be necessary to enter a semi-collapsed building through tiny apertures. This thesis proposes the first quadrotor that can guarantee stable flight independently of the configuration, thanks to an adaptive, morphology-aware control algorithm that continuously optimizes its parameters.

1.4 State of the Art

Tight-coupling of Perception and Action

Coupling perception and action through motion planning and control is not a new problem in robotics. Manipulating the viewpoint of a camera mounted on a robot in order to obtain better perception quality is known in the literature with different names, such as *active vision* [2] or *perception-aware planning*. The literature in this field spans across different typologies of robots, including quadrotors, where coupling perception and action is particularly important due to the limited payload these robots can carry and their underactuated dynamics that needs to be satisfied. From a broad perspective, one can split perception-aware trajectory generation methods into two categories. On the one hand, there are approaches aiming at optimizing the visibility of some points of interest. On the other hand, approaches exist that directly consider the properties of a vision-based algorithm (i.e., Visual-Inertial Odometry, 3D reconstruction, pose estimation) during the planning stage.

Motion Planning for Visibility. The goal of the approaches belonging to this category is to allow a quadrotor to execute a given task, such as following a reference trajectory, while keeping track of some points of interest by ensuring that they lie within the field

of view of a camera mounted onboard the vehicle. In [141], for example, the authors proposed a technique to compute minimum time trajectories for quadrotors that satisfy a limited field of view constraint: the resulting trajectory, therefore, guarantees that the points of interest the robot needs to track are always visible in the image. Perception-aware planning to guarantee visibility is also analyzed in [190], where the authors proposed a differential geometric approach to optimize the trajectory of a quadrotor using a Riemannian manifold. Similarly, the approach proposed in [131] optimizes for the yaw angle of a quadrotor along a reference trajectory consisting of desired positions and derivatives, in order to compute the optimal heading that maximizes the number of features visible for visual localization. [145] and [169] proposed two different solutions to the problem of visual servoing that consider the limitations of the sensor mounted on the robot in order to guarantee that a landmark to be tracked is always visible. In [133] and [134], the authors tackle the problem of generating smooth trajectories for aerial cinematography: the goal is to guarantee that a quadrotor keeps a subject visible at all times while maximizing the quality of the resulting video recordings. Similarly, [182] deals with recording a subject with multiple quadrotors, optimizing the trajectory of each of them in order to produce smooth, nice-looking imagery. Finally, there exist methods that also consider obstacle avoidance when planning perception-aware trajectories, such as the solutions proposed in [146] and [142].

Motion Planning for Improved Algorithm Performance. The second category of perception-aware motion planning approaches directly tries to consider the algorithms that use images as sensing modality, for example vision-based localization, pose estimation, and mapping, in the trajectory generation phase. The main advantage of these approaches is a tighter coupling between perception and action since the trajectory is optimized to maximize the quality of the output of the vision-based algorithm. However, it often comes at the cost of a more complex, often non-convex optimization problem to be solved.

In this regard, there are several examples in the literature of techniques that allow a quadrotor to move in a way that optimizes the performance of vision-based algorithms. In [174] and [57], for example, the authors consider as a task the reconstruction of an environment using structure from motion. In those works, the quality of the map is considered as objective to optimize for, meant as either the accuracy or the uncertainty of the depth estimate. Another common task where perception-aware planning plays a crucial role is state estimation. In [28], the authors proposed a method to plan trajectories that maximize the photometric information in order to guarantee that the vehicle follows the path providing a visual localization algorithm with images containing the highest amount of texture as possible. [194] presented a technique that selects, among many candidates, the best trajectory in terms of a cost function consisting of a combination of probability of collision, progress towards a goal and expected pose estimate error. Similarly, the method shown in [160] minimized the

uncertainty of vision-based state estimation by considering the smallest eigenvalue of the Constructibility Gramian, so that the vehicle can follow the path providing the lowest uncertainty. In [163], the authors proposed an active-vision-based approach to simultaneously estimate the pose of a gap and traverse it.

Low-Latency Sensing and Decision-Making

Sensing latency is due to several factors, including the time necessary to gather each measurement and the frequency of the sensor. Intuitively, the higher the perception latency, the lower the speed a robot can safely navigate through unknown environments. From a thoretical standpoint, the impact of latency on the overall performance of a robotic system has not been intensively investigated. In [69], the authors analyzed the role of the framerate of a camera on the performance of a real-time tracking algorithm. Framerate and latency were also considered in [187], where the authors studied how the parameters of a camera influence the performance of visual servoing. Vision-based navigation in unstructured environments was tackled in [166], which presented an experimental evaluation showing that trading-off framerate, resolution, and latency has an impact on the performance of the system.

Vision-based perception using standard cameras introduces latency due to the exposure time necessary to capture a frame, as well as due to the limited frequency at which it is possible to obtain frames. In order to solve these technological issues, recently researchers investigated the use of neuromorphic sensors, such as event cameras [95, 68], for robotic perception. For example, there exist localization [22, 155, 198] and obstacle detection [161, 119, 113] algorithm based on event cameras which reduce the delay between perception and action. However, the use of low-latency event cameras for closed-loop control is still limited to few simple tasks. For example, the authors of [21] and [127] proposed a technique to control the heading of a robotic platform to track a reference target. A similar task has also been executed with a humanoid robot [65, 66], controlling its gaze to keep track of an object of interest. Other examples are represented by pole balancing [27], goalkeeping [33, 32], and 2D navigation in static environments [25, 63, 13]. More complex systems, such as quadrotors, have been used for the tasks of avoiding dynamic obstacles in [126] and [162], using event cameras to detect incoming objects.

Adaptive Morphology for Enhanced Maneuverability

Adaptive morphology [177] recently started drawing the attention of the robotics community. The ability to change the mechanical properties of a robotic platform, such as its shape, can lead to essential advantages in terms of locomotion, maneuverability, and versatility. For example, by suitably adapting its shape, a robot can be able to both fly and walk [14, 30], unlocking the potential of both locomotion modalities

within the same system. A non-rigid mechanical structure is particularly effective in increasing the resilience of flying robots, also allowing new transportation and deployment methods [170, 118, 116]. Additionally, a number of solutions based on the use of tiltable rotors [82, 158, 157] have been proposed to overcome the limitations of the underactuated dynamics of a quadrotor, improving the maneuverability of these platforms.

One of the main benefits of morphing robots is the possibility to adapt their size and shape to a particular scenario or task. For instance, it is possible to let a flying robot negotiate tight spaces, narrower than its size, by adjusting its mechanical structure [196, 35, 153, 197, 19], transport objects by wrapping around them [195], or modify the robot's dynamics in order to improve trajectory tracking [6, 159], deal with rotor failure [5], optimize energy consumption [193] or obtain novel maneuvering modalities [185]. The main challenge in developing shape-shifting quadrotors is the design of a control strategy that allows the robot to quickly change its morphology in flight while guaranteeing stability at all times. Most of the approaches proposed in the literature have only been shown in simulation [189, 5], require long morphing times [196], or cannot provide stable flight in morphologies different from the standard X [153, 35, 19]. For this reason, in [50] I presented the foldable drone, the first quadrotor that can actively change its shape to perform different tasks. For example, it can traverse gaps that are smaller than its size, transport objects and inspect surfaces from very close. The system does not require any symmetry in the morphology to ensure stable flight.

1.5 Summary

In this chapter, I discussed how quadrotors work, why they play a crucial role in the robotics revolution, and what are the main challenges in autonomous, agile flight. In a literature research, I summarized the state of the art of tightly coupled perception and action, low-latency sense-and-avoid, and adaptive morphology for quadrotors. Furthermore, I summarized the objectives of this work and how they relate and build upon state of the art.

2 Contributions

This chapter summarizes the key contributions of the papers that are reprinted in the appendix. It further highlights the connections between the individual results and refers to related video and open-source code contributions. In total, this research has been published in five peer-reviewed conference publications and six journal publications. One further journal paper is currently under review at *AAAS Science Robotics*. Some of these works, such as B and E, were also successfully demonstrated live in multiple countries around the world. Additionally, the work E was awarded the 2019 *Drone Hero Award* for the category "Most Innovative Drone" and the Tech Briefs 2019 *Create the Future* first prize for the category "Aerospace & Defense".

2.1 Tightly Coupled Perception and Action

In this section, the contributions for tightly coupled perception and action are listed. These works showed that by considering perception when planning trajectories for quadrotors, aggressive maneuvers can be executed without relying on external positioning systems, but rather only relying on cameras.

2.1.1 Paper A: Aggressive Flight through Narrow Gaps

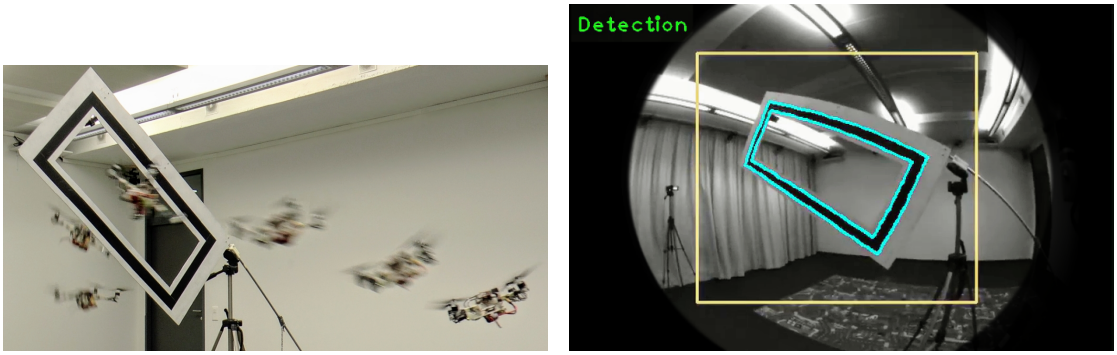
(P1) D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza. "Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017. DOI: 10.1109/icra.2017.7989679

We address one of the biggest challenges towards autonomous quadrotor flight in complex environments, which is flight though narrow gaps. For this, we equipped a quadrotor with a front-looking camera, an inertial measurement unit, and an onboard computer to autonomously detect a gap and traverse it by only using onboard sensing and computing. We estimate the quadrotor's state by computing its relative pose to the gap from the captured images and fuse it with measurements from the inertial measurement unit. We then compute a trajectory that enables the quadrotor to safely pass narrow, inclined gaps with an agile maneuver. Our method generates a trajectory

that considers geometric, dynamic, and perception constraints: during the approach maneuver, the quadrotor always faces the gap to allow state estimation, while respecting the vehicle dynamics; during the traverse through the gap, the distance of the quadrotor to the edges of the gap is maximized. We re-plan the trajectory during its execution to cope with the varying uncertainty of the state estimate. In real experiments, we demonstrate a success rate of 80 % for gap inclinations of up to 45° with out approach.

Related Videos

(V1) https://youtu.be/meSItatXQ7M



(a) The quadrotor passing through the gap **(b)** View from the onboard camera

Figure 2.1: (a) Sequence of our quadrotor passing through a narrow, 45°-inclined gap. Our state estimation fuses gap detection from a single onboard forward-facing camera (b) with an IMU. All planning, sensing, control run fully onboard a smartphone computer.

2.1.2 Paper B: Perception-Aware Model Predictive Control

(P2) D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza. "PAMPC: Perception-Aware Model Predictive Control for Quadrotors". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Oct. 2018. DOI: 10.1109/IROS.2018.8593739

We present the first perception-aware model predictive control framework for quadrotors that unifies control and planning with respect to action and perception objectives. Our framework leverages numerical optimization to compute trajectories that satisfy the system dynamics and require control inputs within the limits of the platform. Simultaneously, it optimizes perception objectives for robust and reliable sensing by maximizing the visibility of a point of interest and minimizing its velocity in the image plane. Considering both perception and action objectives for motion planning and control is challenging due to the possible conflicts arising from their respective requirements. For example, for a quadrotor to track a reference trajectory, it needs to rotate to align its thrust with the direction of the desired acceleration. However, the perception objective might require to minimize such rotation to maximize the visibility of a point of interest. A model-based optimization framework, able to consider both

perception and action objectives and couple them through the system dynamics, is therefore necessary. Our perception-aware model predictive control framework works in a receding-horizon fashion by iteratively solving a non-linear optimization problem. It is capable of running in real-time, fully onboard our lightweight, small-scale quadrotor using a low-power ARM computer, together with a visual-inertial odometry pipeline. We validate our approach in experiments demonstrating (i) the conflict between perception and action objectives, and (ii) improved behavior in extremely challenging lighting conditions.

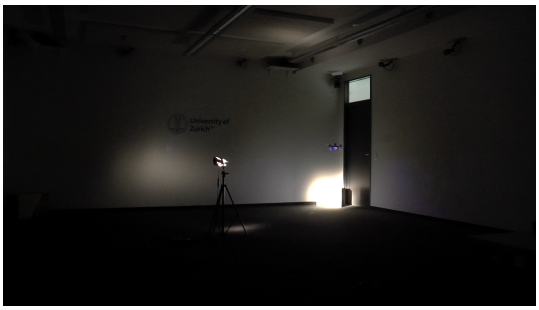
Related Videos

(V2) https://youtu.be/9vaj829vE18

Related Software

(S1) https://github.com/uzh-rpg/rpg_mpc

(a) The quadrotor flying along a circle trajectory, while tracking the boxes in the center thanks to perception term in the cost function.



(b) The quadrotor flying in a dark environment, looking at the only illuminated spot to localize itself.

Figure 2.2: (a) Sequence of our quadrotor flying at $3\,\mathrm{m\,s^{-1}}$ along a circular trajectory and simultaneously keeping visible in the image the visual features provided by the boxes in the center of the room. Our algorithm allows to follow reference trajectories while improving visual perception, as for example in a dark room (b) where only a small region provides sufficient visual information in order to perform reliable localization.

2.2 Low-Latency Perception to Action

In this section, the contributions for low-latency perception and decision making are listed. These contributions are both theoretical and practical: on the one hand, the problem of studying the impact of perception latency on high-speed sense and avoid is studied; on the other, a reactive, low-latency framework to avoid fast dynamic obstacles is presented.

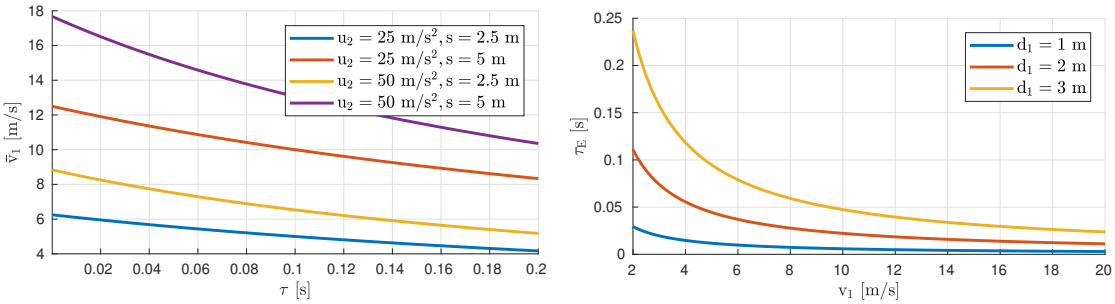
2.2.1 Paper C: The Role of Perception Latency in Obstacle Avoidance

(P3) D. Falanga, S. Kim, and D. Scaramuzza. "How Fast is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid". In: *IEEE Robot. Autom. Lett.* 4.2 (Apr. 2019), pp. 1884–1891. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2898117

In this work, we study the effects that perception latency has on the maximum speed a robot can reach to safely navigate through an unknown cluttered environment. We provide a general analysis that can serve as a baseline for future quantitative reasoning for design trade-offs in autonomous robot navigation. We consider the case where the robot is modeled as a linear second-order system with bounded input and navigates through static obstacles. We show how the maximum latency that the robot can tolerate to guarantee safety is related to the desired speed, the range of its sensing pipeline, and the actuation limitations of the platform (i.e., its agility, measured as the maximum acceleration it can produce). As a particular case study, we compare monocular and stereo frame-based cameras against novel, low-latency sensors, such as event cameras, in the case of quadrotor flight. To the best of our knowledge, this is the first theoretical work in which perception and actuation limitations are jointly considered to study the performance of a robotic platform in high-speed navigation.

Related Videos

(V3) https://youtu.be/sbJAi6SXOQw



(a) Maximum safe travel speed $\bar{v}_1$ for a robot, in the case of obstacle size $r = 0.5$ m.

(b) The theoretical latency τ_E for an event camera with QVGA resolution.

Figure 2.3: (a) The maximum speed that a robot can achieve in order to safely navigate through static obstacles. This maximum speed is function of its perception latency, which in (b) is reported for an event camera.

2.2.2 Paper D: Event-based avoidance

(P4) D. Falanga, K. Kleber, and D. Scaramuzza. "Low Latency Avoidance of Dynamic Obstacles for Quadrotors with Event Cameras". In: *AAAS Science Robotics, Under Review* (2019)

In this paper, we address one of the fundamental challenges for micro aerial vehicles:

dodging fast moving objects using only onboard sensing and computation. Effective avoidance of moving obstacles requires fast reaction times, which entails low-latency sensors and algorithms for perception and decision making. All existing works rely on standard cameras, which have latencies of tens of milliseconds and suffer from motion blur. We depart from state of the art by relying on a novel bioinspired sensor, called event camera, with reaction times of microseconds, which perfectly fits our task requirements. However, because the output of this sensor is not images but a stream of asynchronous events that encode per-pixel intensity changes, standard vision algorithms cannot be applied. Thus, a paradigm shift is necessary to unlock the full potential of event cameras. Our proposed framework exploits the temporal information contained in the event stream to distinguish between static and dynamic objects, and makes use of a fast strategy to generate the motor commands necessary to avoid the detected obstacles. Our resulting algorithm has an overall latency of only 3.5 ms, which is sufficient for reliable detection and avoidance of fast-moving obstacles. We demonstrate the effectiveness of our approach on an autonomous quadrotor avoiding multiple obstacles of different sizes and shapes, at relative speeds up to $10\,\mathrm{m\,s^{-1}}$, both indoors and outdoors.

Related Videos

(V4) http://rpg.ifi.uzh.ch/event_based_avoidance

Figure 2.4: Sequence of an avoidance maneuver.

2.3 Morphing Quadrotors

This section presents the foldable drone, the first quadrotor that is capable of changing shape and size while flying without compromising the stability of the vehicle.

2.3.1 Paper E: The Foldable Drone

(P5) D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza. "The Foldable Drone: A Morphing Quadrotor that can Squeeze and Fly". In: *IEEE Robot. Autom. Lett.* 4.2 (Apr. 2019), pp. 209–216. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2885575

The recent advances in state estimation, perception, and navigation algorithms have significantly contributed to the ubiquitous use of quadrotors for inspection, mapping,

and aerial imaging. To further increase the versatility of quadrotors, recent works investigated the use of an adaptive morphology, which consists of modifying the shape of the vehicle during flight to suit a specific task or environment. However, these works either increase the complexity of the platform or decrease its controllability. In this paper, we propose a novel, simpler, yet effective morphing design for quadrotors consisting of a frame with four independently rotating arms that fold around the main frame. To guarantee stable flight at all times, we exploit an optimal control strategy that adapts on the fly to the drone morphology. We demonstrate the versatility of the proposed adaptive morphology in different tasks, such as negotiation of narrow gaps, close inspection of vertical surfaces, and object grasping and transportation. The experiments are performed on an actual, fully autonomous quadrotor relying solely on onboard visual-inertial sensors and compute. No external motion tracking systems and computers are used. This is the first work showing stable flight without requiring any symmetry of the morphology.

Related Videos

(V5) https://youtu.be/jmKXCdEbF_E

(a) Different morphologies: H, to traverse vertical gaps; O, to pass through horizontal gaps; T, to inspect surfaces.

(b) A sequence showing the foldable drone traversing a vertical, narrow gap assuming the H morphology.

Figure 2.5: Adaptive morphology (a) allows a quadrotor to execute tasks that are not possible for a vehicle with a fixed morphology. For example, it can traverse gaps (b) that are smaller than its size, by assuming a suitable shape that lets it squeeze through them.

2.4 Applications of Vision-Based Quadrotors

This section presents the contributions of this thesis in terms of applications of vision-based quadrotors. In particular, it contains a paper that is the result of the work towards the MBZIRC 2017 competition, where the Robotics and Perception Group participated in the Challenge 1, requiring an autonomous quadrotor to land on a moving platform.

2.4.1 Paper F: Autonomous Landing on a Moving Platform

(P6) D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza. "Vision-based Autonomous Quadrotor Landing on a Moving Platform". In: *IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*. Oct. 2017. DOI: 10.1109/SSRR.2017.8088164

We present a quadrotor system capable of autonomously landing on a moving platform using only onboard sensing and computing. We rely on state-of-the-art computer vision algorithms, multi-sensor fusion for localization of the robot, detection and motion estimation of the moving platform, and path planning for fully autonomous navigation. Our system does not require any external infrastructure, such as motion-capture systems. No prior information about the location of the moving landing target is needed. We validate our system in both synthetic and real-world experiments using low-cost and lightweight consumer hardware. To the best of our knowledge, this is the first demonstration of a fully autonomous quadrotor system capable of landing on a moving target, using only onboard sensing and computing, without relying on any external infrastructure.

Related Videos

(V6) https://youtu.be/Tz5ubwoAfNE

2.5 Unrelated Contributions

During the Ph.D., six papers were co-authored that are not part of the Ph.D. work itself. These papers have as main topic the development of algorithms for motion planning and control of quadrotors for high-speed, agile flight.

(U1) P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza. "Fast Trajectory Optimization for Agile Quadrotor Maneuvers with a Cable-Suspended Payload". In: *Robotics: Science and Systems (RSS)*. June 2017. DOI: 10.15607/RSS.2017.XIII.030

(U2) M. Faessler, D. Falanga, and D. Scaramuzza. "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight". In: *IEEE Robot. Autom. Lett.* 2.2 (Apr. 2017), pp. 476–482. ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2640362

(U3) R. Spica, D. Falanga, E. Cristofalo, E. Montijano, D. Scaramuzza, and M. Schwager. "A Game Theoretic Approach to Autonomous Two-Player Drone Racing". In: *Robotics: Science and Systems (RSS)*. June 2018. DOI: 10.15607/RSS.2018.XIV.040

(U4) S. Kim, D. Falanga, and D. Scaramuzza. "Computing the Forward Reachable Set for a Multirotor Under First-Order Aerodynamic Effects". In: *IEEE Robot. Autom. Lett.* 3.4 (Oct. 2018), pp. 2934–2941. DOI: 10.1109/LRA.2018.2848302

(U5) B. Nisar, P. Foehn, D. Falanga, and D. Scaramuzza. ""VIMO: Simultaneous Visual Inertial Model-based Odometry and Force Estimation". In: *IEEE Robot. Autom. Lett.* 4.3 (July 2019), pp. 2785–2792. DOI: 10.1109/LRA.2019.2918689

(U6) H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, G. de Croon, S. Hwang, S. Jung, H. Shim, H.

Kim, M. Park, T.-C. Au, and S. J. Kim. “Challenges and implemented technologies used in autonomous drone racing”. In: *Intelligent Service Robotics* 12.2 (Apr. 2019), pp. 137–148. ISSN: 1861-2784. DOI: 10.1007/s11370-018-00271-6

3 Future Directions

When this doctoral work started, quadrotors were mostly a research platform. The number of companies manufacturing drones was significantly lower than today, and the capabilities of these vehicles were limited most of the time to manual flight within line-of-sight. The past four years, however, have seen tremendous progress in the field of aerial robotics, both in terms of off-the-shelf availability of quadrotors (one find systems capable of some sort of autonomous flight in almost any electronics shop) and autonomy of these robots. The recent advances in autonomous vision-based quadrotor flight, highlighted by the impressive performance of the Skydio R1, recently led researcher to question whether aerial robotics is still an open research area, and what are the next challenges that academia should try to face in order to stay ahead of industry [1]. Nevertheless, before considering aerial robotics a solved problem, there exist several technological and scientifical challenges to be tackled.

In this section, I will discuss the main limitations of the approaches I proposed to tackle the research questions analyzed in this doctoral thesis, and will provide some possible future search directions.

3.1 Limitations of the Proposed Approaches

Tight Coupling of Perception and Action. In my thesis, I tackled the problem of coupling perception and action using two different approaches, namely by exploiting both sampling-based [46] and optimization-based [49] methods. Sampling-based methods do not guarantee optimality of the selected trajectory, and since they require fast methods to compute candidate trajectories for each sample, it might be not possible to guarantee feasibility in terms of inputs saturation. These issues can be solved by adopting an optimization-based approach, which however struggle with dealing with non-convex scenarios, where sampling-based methods instead can provide added value. Therefore, both approaches have their own limitations, which might be overcome by combining them: for example, one could use sampling-based planning to obtain a candidate reference trajectory to initialize an optimization-based planning algorithm.

[1] http://www.seas.upenn.edu/~loiannog/workshopIROS2018uav/

Low-Latency Sensing and Decision-Making. In [47] I analized the impact of sensing latency on the speed a robot can achieve to navigate safely in an unknown environments, comparing standard cameras to event cameras. The main limitation of that approach is the static environment assumption, where dynamic obstacles are expected to significantly increase the advantages of event cameras against standard cameras thanks to the fact that they are *motion-activated* sensors, and therefore are the perfect fit for sensing moving objects. Additionally, the mathematical analysis performed makes use of a second-order model for the robot, without considering any dynamics on the inputs. Extending the same considerations to the case where the inputs cannot be applied instantenously, but rather have their own dynamics, would potentially increase the level of details of the aforementioned analys.

Adaptive Morphology for Enhanced Maneuverability. The morphing approach I proposed in [50] allows a quadrotor to change its shape by folding the arms around the main body. However, this comes at the cost of increasing the mechanical complexity of the system, as well as its weight, in order to accommodate the additional hardware required for changing morphology. Furthermore, the folding mechanism used in [50] is particularly fragile, and suffers from crashes in real-world experiments. This can potentially render the system unstable due to unexpected vibrations deriving from damages to the servo-motors. Future work should focus on more robust, potentially passive, folding mechanism, in order to increase the effectiveness of the proposed approach. Additionally, the control scheme in [50] does not take into account the effects of the morphing on the thrust produced by each motor, which future work should consider in order to improve the tracking capabilities of the vehicle.

3.2 Future Work

Task representation. There exist several ways of formalizing a task in terms of perception, planning, and control. Different representations can bring different advantages and disadvantages. For example, a representation that simplifies perception could render trajectory planning and control more complicated. Similarly, representing the same task in a way that makes trajectory planning easier could be not suited for vision-based perception. As a concrete example, consider the task of traversing a narrow gap. The approach I proposed in [46] simplifies perception, since only a gap detector is sufficient, but complicates trajectory planning by introducing a *perception-awareness* requirement to make the gap visible at all times. [99] tackled the same task in a way that simplifies planning, since a single trajectory is computed and executed. However, it makes perception more cumbersome since it relies on Visual-Inertial Odometry and, to account for the drift in the localization, it would require an additional perception module to detect the gap. This example shows how different representations for the same task can impact the different modules that compose a robotic system, and the problem

of choosing the best representation for the task at hand, simultaneously considering perception, planning, and control, is still unsolved.

Tight coupling of perception and action. At the moment, most of the techniques for perception-aware motion planning and control follow either one of the two approaches reported in Sec. 1.4. However, a method that simultaneously understands the impact of a given action on the quality of Visual-Inertial localization and renders the region providing the highest visual information visible at all times is still missing. The development of a real-time motion planning technique capable of doing so would significantly increase the performance of vision-based closed-loop flight since it would enhance the robustness of the localization system and the quality of its output. The main challenge in developing such a system is formulating the expected quality of vision-based localization (i.e., uncertainty, drift, the robustness of the features available) along a candidate trajectory, based on the visual information currently available. Additionally, such a formulation should be suitable for numerical optimization. Namely, it should satisfy properties such as convexity and differentiability, and be usable in a real-time optimization framework to allow fast re-planning.

Event-based control. As shown in Sec. 1.4, the exist a number of works that exploit event cameras for low-latency control. However, the majority of these works accumulate events for a given amount of time and process all the accumulated events simultaneously. In other words, most of the time, event cameras are used as edge-triggered, high-frequency cameras, without exploiting the asynchronous nature of this sensor. There is, therefore, a lack of asynchronous control algorithms that can exploit events as they arrive at the processing unit, without waiting for them to be accumulated and processed. Such control algorithms would depart from the standard approach of exploiting feedback at a known rate and would have the critical advantage of being able to exploit partial information (i.e., few events) to reduce the latency between perception and action significantly. In this regard, the biggest challenge is represented by determining how much information (i.e., how many events) is necessary for robust and reliable inference. Indeed, single events do not provide information, and accumulating too many events would lead to a synchronous use of event cameras.

Morphing design. Morphing can significantly enhance the capabilities and maneuverability of quadrotors. However, my work [50] still represents the only example of shape-shifting multirotor capable of guaranteeing stable flight with any morphology. Morphing capabilities are destined to become standard on future quadrotors. Nevertheless, this field is mostly unexplored and offers excellent potentials in terms of: (i) novel applications, since the possibility of changing shape and size while flying can unlock the execution of tasks that are not possible with fixed-morphology drones;

(ii) mechanical designs, investigating lightweight, crash-resilient and energy-efficient morphing strategies; (iii) motion planning and control, where morphology awareness is necessary to guarantee stability but can also be exploited to improve maneuverability and efficiency.

Summary

Quadrotors are extremely powerful flying machines, offering enormous potentials thanks to their agility and maneuverability. However, as of today, their potential is only partially exploited. The main challenges towards a better usage of these capabilities lie at the intersection between perception and action, given the need for algorithms that reliably and effectively consider the limitations of vision-based perception (motion blur, lack of texture, latency) for localization and sensing. Additionally, morphing offers the chance to boost the maneuverability of a flying robot and the range of tasks it can execute.

A Aggressive Flight through Narrow Gaps

D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza. "Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017. DOI: 10.1109/icra.2017.7989679

Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing using Active Vision

Davide Falanga, Elias Mueggler, Matthias Faessler and Davide Scaramuzza

Abstract — We address one of the main challenges towards autonomous quadrotor flight in complex environments, which is flight through narrow gaps. While previous works relied on off-board localization systems or on accurate prior knowledge of the gap position and orientation in the world reference frame, we rely solely on onboard sensing and computing and estimate the full state by fusing gap detection from a single onboard camera with an IMU. This problem is challenging for two reasons: (i) the quadrotor pose uncertainty with respect to the gap increases quadratically with the distance from the gap; (ii) the quadrotor has to actively control its orientation towards the gap to enable state estimation (i.e., active vision). We solve this problem by generating a trajectory that considers geometric, dynamic, and perception constraints: during the approach maneuver, the quadrotor always faces the gap to allow state estimation, while respecting the vehicle dynamics; during the traverse through the gap, the distance of the quadrotor to the edges of the gap is maximized. Furthermore, we replan the trajectory during its execution to cope with the varying uncertainty of the state estimate. We successfully evaluate and demonstrate the proposed approach in many real experiments, achieving a success rate of 80% and gap orientations up to 45°. To the best of our knowledge, this is the first work that addresses and achieves autonomous, aggressive flight through narrow gaps using only onboard sensing and computing and without prior knowledge of the pose of the gap.

Supplementary Material

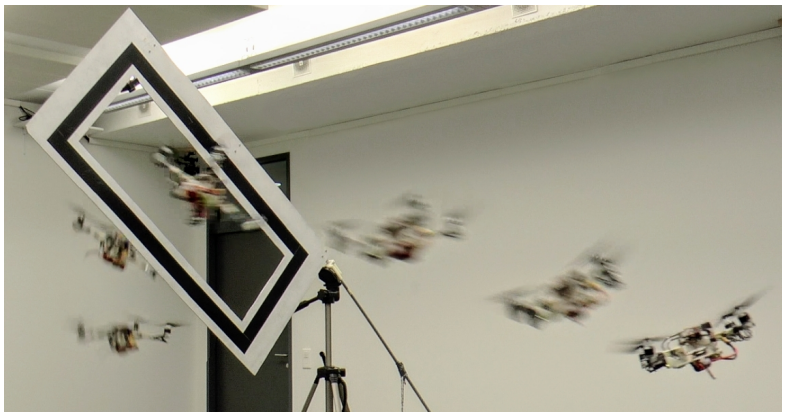
The accompanying video is available at:
http://rpg.ifi.uzh.ch/aggressive_flight.html

A.1 Introduction

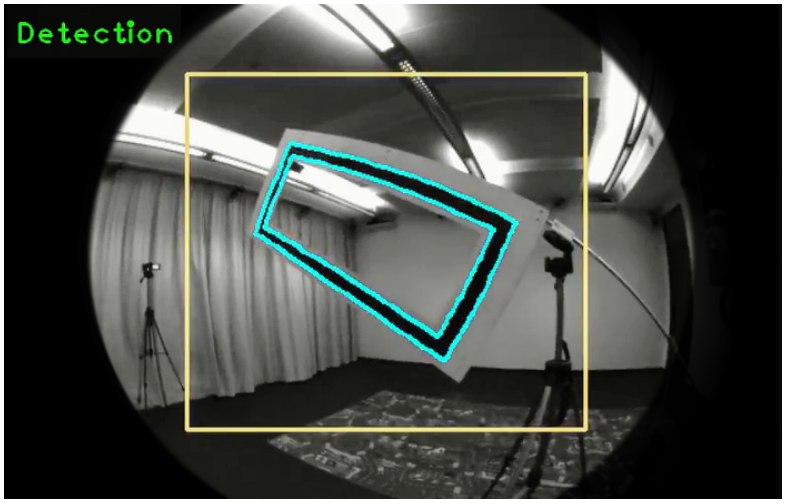
Recent works have demonstrated that micro quadrotors are extremely agile and versatile vehicles, able to execute very complex maneuvers [128, 29, 112]. These demonstrations highlight that one day quadrotors could be used in search and rescue applications, such as in the aftermath of an earthquake, to navigate through buildings, by entering and exiting through narrow gaps, and to quickly localize victims.

In this paper, we address one of the main challenges towards autonomous quadrotor flight in complex environments, which is flight through narrow gaps. What makes this problem challenging is that the gap is very small, such that precise trajectory-following is required, and can be oriented arbitrarily, such that the quadrotor cannot fly through it in near-hover conditions. This makes it necessary to execute an aggressive trajectory (i.e., with high velocity and angular accelerations) in order to align the vehicle to the gap orientation (cf. Fig. A.1).

Previous works on aggressive flight through narrow gaps have focused solely on the control and planning problem and therefore relied on accurate state estimation from external motion-capture systems and/or accurate knowledge of the gap position and orientation in the world reference frame. Since these systems were not gap-aware, the trajectory was generated before execution and never replanned. Therefore, errors in the measure of the pose of the gap in the world frame were not taken into account, which may lead to a collision with gap. Conversely, we are interested in using only *onboard sensing and computing*, *without any prior knowledge* of the gap pose in the world frame. More specifically, we address the case where state estimation is done by fusing gap detection through a single, forward-facing camera with an IMU. We show that this raises an interesting *active-vision* problem (i.e, coupled perception and control). Indeed, for the robot to localize with respect to the gap, a trajectory that guarantees that the quadrotor always faces the gap must be selected (perception constraint). Additionally, it must be replanned multiple times during its execution to cope with the varying uncertainty of the state estimate, which is quadratic with the distance from the gap. Furthermore, during the traverse, the quadrotor must maximize the distance from the edges of the gap (geometric constraint) to avoid collisions. At the same time, it must do so without relying on any visual feedback (when the robot is very close to the gap, it exits from the field of view of the camera). Finally, the trajectory must be feasible with respect to the dynamic constraints of the vehicle.



(a) The quadrotor passing through the gap.



(b) View from the onboard camera

Figure A.1: Sequence of our quadrotor passing through a narrow, 45°-inclined gap. Our state estimation fuses gap detection from a single onboard forward-facing camera with an IMU. All planning, sensing, control run fully onboard on a smartphone computer.

Our proposed trajectory generation approach is independent of the gap-detection algorithm being used; thus, to simplify the perception task, we use a gap with a black-and-white rectangular pattern (cf. Fig. A.1) for evaluation and demonstration.

A.1.1 Related Work

A solution for trajectory planning and control for aggressive quadrotor flight was presented in [112]. The authors demonstrated their results with aggressive flight through a narrow gap, and by perching on inclined surfaces. The quadrotor state was obtained using a motion-capture system.

To fly through a narrow gap, the vehicle started by hovering in a pre-computed position,

flew a straight line towards a launch point, and then controlled its orientation to align with the gap. The method was not *plug-and-play* since it needed training through *iterative learning* in order to refine the launch position and velocity. This was due to the instantaneous changes in velocity caused by the choice of a straight line for the approach trajectory. Unlike their method, we use a technique that computes polynomial trajectories which are guaranteed to be feasible with respect to the control inputs. The result is a smooth trajectory, compatible with the quadrotor dynamic constraints, which makes learning unnecessary. Indeed, in realistic scenarios, such as search-and-rescue missions, we cannot afford training but must pass on the first attempt.

In [110], the same authors introduced a method to compute trajectories for a quadrotor solving a Quadratic Program, which minimizes the snap (i.e., the fourth derivative of position). In their experiments, agile maneuvers, such as passing through a hula-hoop thrown by hand in the air, were demonstrated using state estimation from a motion-capture system.

In [184], a technique that lets a quadrotor pass through a narrow gap while carrying a cable-suspended payload was presented and was experimentally validated using a motion-capture system for state estimation.

In [136], the authors proposed an unconstrained nonlinear model predictive control algorithm in which trajectory generation and tracking are treated as a single, unified problem. The proposed method was validated in a number of experiments, including a rotorcraft passing through an inclined gap. Like the previous systems, they used a motion-capture system for state estimation.

In [105], the authors proposed a vision-based method for autonomous flight through narrow gaps by fusing data from a downward and a forward-looking camera, and an IMU. Trajectory planning was executed on an external computer. However, the authors only considered the case of an horizontal gap, therefore no agile maneuver was necessary.

In [99], the authors proposed methods for onboard vision-based state estimation, planning, and control for small quadrotors, and validated the approach in a number of agile maneuvers, among which flying through an inclined gap. Since state estimation was performed by fusing input from a downward-looking camera and an IMU, rather than from gap detection, the gap position and orientation in the world reference frame had to be measured very accurately prior to the execution of the maneuver. The trajectory was generated before execution and never replanned. Therefore, errors in the measure of the pose of the gap in the world frame were not taken into account, which may lead to a collision with gap. To deal with this issue, the authors used a gap considerably larger than the vehicle size.

All the related works previously mentioned relied on the accurate state estimates

from a motion-capture system or accurate prior knowledge of the gap position and orientation in the world reference frame. Additionally, in all these works but [136] and [99] trajectory generation was performed on an external computer. The advantages of a motion-capture system over onboard vision are that the state estimate is always available, at high frequency, accurate to the millimeter, and with almost *constant noise covariance* within the tracking volume. Conversely, a state estimate from onboard vision can be intermittent (e.g., due to misdetections); furthermore, its covariance increases *quadratically* with the distance from the scene and is strongly affected by the type of structure and texture of the scene. Therefore, to execute a complex aggressive maneuver, like the one tackled in this paper, while using only onboard sensing and *gap-aware* state estimation, it becomes necessary *to couple perception with the trajectory generation process* (i.e., active vision). Specifically, the desired trajectory has to render the gap always visible by the onboard camera in order to estimate its relative pose.

A.1.2 Contributions

Our method differs from previous works in the following aspects: (i) we rely solely on onboard, visual-inertial sensors and computing, (ii) we generate a trajectory that facilitates the perception task, while satisfying geometric and dynamic constraints, and (iii) we do not require iterative learning, neither do we need to know a priori the gap position and orientation in the world frame. To the best of our knowledge, this is the first work that addresses and achieves aggressive flight through narrow gaps with state estimation via gap detection from an onboard camera and IMU.

The remainder of this paper is organized as follows. Section A.2 presents the proposed trajectory-generation algorithm. Section A.3 describes the state-estimation pipeline. Section A.4 presents the experimental results. Section A.5 discusses the results and provides additional insights about the approach. Finally, Section A.6 draws the conclusions.

A.2 Trajectory Planning

We split the trajectory planning into two consecutive stages. First, we compute a traverse trajectory to pass through the gap. This trajectory maximizes the distance from the vehicle to the edges of the gap in order to minimize the risk of collision. In a second stage, we compute an approach trajectory in order to fly the quadrotor from its current hovering position to the desired state that is required to initiate the traverse trajectory. While both trajectories need to satisfy *dynamic constraints*, the approach trajectory also satisfies *perception constraints*, i.e., it lets the vehicle-mounted camera always face the gap. This is necessary to enable state estimation with respect to the gap.

A.2.1 Traverse Trajectory

During the gap traversal, the quadrotor has to minimize the risk of collision. We achieve this by forcing the traverse trajectory to intersect the center of the gap while simultaneously lying in a plane orthogonal to the gap (see Fig. A.2). In the following, we derive the traverse trajectory in this orthogonal plane and then transform it to the 3D space.

Let W be our world frame. The vector $\mathbf{p}_G$ and the rotation matrix $\mathbf{R}_G$ denote the position of the geometric center of the gap and its orientation with respect to W, respectively. Let Π be a plane orthogonal to the gap, passing through its center and parallel to the longest side of the gap (cf. Fig. A.2). Let $\mathbf{e}_1$ and $\mathbf{e}_2$ be the unit vectors spanning such a plane Π, whose normal unit vector is $\mathbf{e}_3$. The $\mathbf{e}_2$ axis is orthogonal to the gap and $\mathbf{e}_1 = \mathbf{e}_2 \times \mathbf{e}_3$.

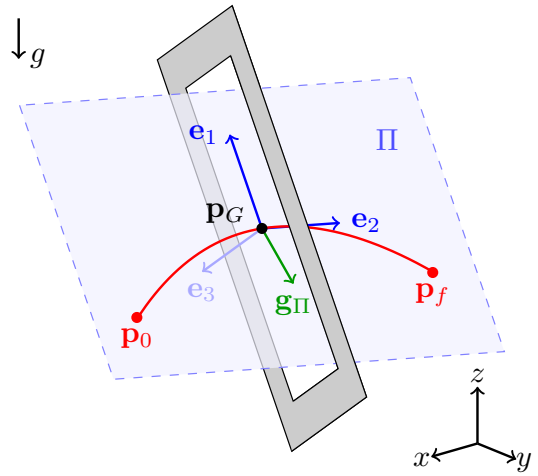


Figure A.2: An inclined gap and the corresponding plane Π.

Intuitively, a trajectory that lies in the plane Π and passes through the center of the gap, minimizes the risk of impact with the gap.

To constrain the motion of the vehicle to the plane Π, it is necessary to compensate the projection of the gravity vector $\mathbf{g}$ onto its normal vector $\mathbf{e}_3$. Therefore, a constant thrust of magnitude $\langle \mathbf{g}, \mathbf{e}_3 \rangle$ needs to be applied orthogonally to Π. By doing this, a 2D description of the quadrotor's motion in this plane is sufficient. The remaining components of $\mathbf{g}$ in the plane Π are computed as

$$\mathbf{g}_\Pi = \mathbf{g} - \langle \mathbf{g}, \mathbf{e}_3 \rangle \, \mathbf{e}_3. \tag{A.1}$$

Since this is a constant acceleration, the motion of the vehicle along Π is described by

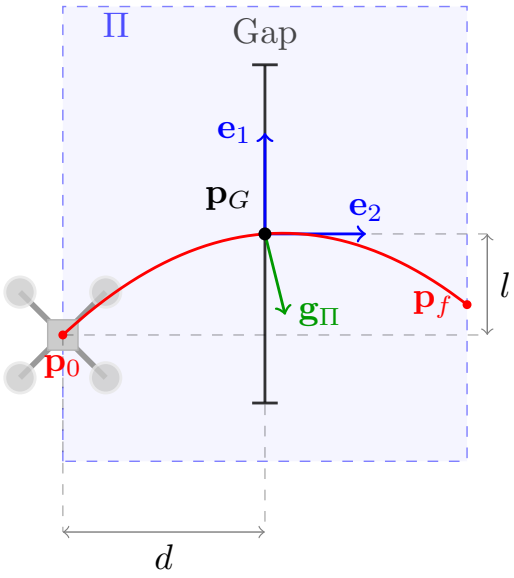


Figure A.3: The traverse trajectory in the plane Π.

the following second order polynomial equation:

$$p_i(t) = p_i(t_0) + v_i(t_0)t + \frac{1}{2}g_{\Pi,i}t^2, \tag{A.2a}$$

$$v_i(t) = v_i(t_0) + g_{\Pi,i}t, \tag{A.2b}$$

where the subscript $i = \{1, 2\}$ indicates the component along the $\mathbf{e}_i$ axis. The quadrotor enters the traverse trajectory at time t_0, t is the current time, and p and v denote its position and velocity, respectively.

Equation (A.2) describes a ballistic trajectory. When $g_{\Pi,2} = 0$, it is the composition of a uniformly accelerated and a uniform-velocity motion. In other words, in these cases the quadrotor moves on a parabola in space.

Let l and d be the distance between $\mathbf{p}_G$ and the initial point of the trajectory, $\mathbf{p}_0$, along $\mathbf{e}_1$ and $\mathbf{e}_2$, respectively (cf. Fig. A.3). These two parameters determine the initial position and velocity in the plane Π, as well as the time t_c necessary to reach $\mathbf{p}_G$. The values of d and l are determined through an optimization problem, as explained later in Sec. A.2.2.

For a generic orientation $\mathbf{R}_G$ of the gap, (A.2) is characterized by a uniformly accelerated motion along both the axes $\mathbf{e}_1$ and $\mathbf{e}_2$. Therefore, it is not possible to guarantee that the distance traveled along the $\mathbf{e}_2$ axis before and after the center of the gap are equal while also guaranteeing that the initial and final position have the same coordinate along the $\mathbf{e}_1$ axis. For safety reasons, we prefer to constrain the motion along the $\mathbf{e}_2$ axes, i.e., orthogonally to the gap, such that the distances traveled before and after the

gap are equal.

Given the components of the unit vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ in the world frame, it is now possible to compute the initial conditions $\mathbf{p}_0 = \mathbf{p}(t_0)$ and $\mathbf{v}_0 = \mathbf{v}(t_0)$ in 3D space as follows:

$$\mathbf{p}_0 = \mathbf{p}_G - l\mathbf{e}_1 - d\mathbf{e}_2, \tag{A.3a}$$

$$\mathbf{v}_0 = \left(\frac{l}{t_c} - \frac{1}{2}g_{\Pi,1}t_c\right)\mathbf{e}_1 + \left(\frac{d}{t_c} - \frac{1}{2}g_{\Pi,2}t_c\right)\mathbf{e}_2, \tag{A.3b}$$

where:

$$t_c = \sqrt{\frac{-2l}{g_{\Pi,1}}} \tag{A.4}$$

is the time necessary to reach the center of the gap once the traverse trajectory starts.

Note that this solution holds if $g_{\Pi,2} \geq 0$ which applies if $\mathbf{e}_2$ is horizontal or pointing downwards in world coordinates. The case $g_{\Pi,2} < 0$ leads to similar equations, which we omit for brevity. The final three-dimensional trajectory then has the following form:

$$\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}_0 t + \frac{1}{2}\mathbf{g}_\Pi t^2, \tag{A.5a}$$

$$\mathbf{v}(t) = \mathbf{v}_0 + \mathbf{g}_\Pi t, \tag{A.5b}$$

$$\mathbf{a}(t) = \mathbf{g}_\Pi. \tag{A.5c}$$

This trajectory is inexpensive to compute since it is solved in closed form. Also, note that during the traverse the gap is no longer detectable. Nevertheless, since the traverse trajectory is short and only requires *constant control inputs* (a thrust of magnitude $\langle \mathbf{g}, \mathbf{e}_3 \rangle$ and zero angular velocities), it is possible to track it accurately enough to not collide with the gap, even without any visual feedback.

A.2.2 Optimization of the Traverse Trajectory

To safely pass through the gap, the quadrotor must reach the initial position and velocity of the traverse trajectory described by (A.3a)-(A.3b) with an acceleration equal to $\mathbf{g}_\Pi$ at time t_0. An error in these initial conditions is propagated through time according to (A.5a)-(A.5c), and therefore may lead to a collision. The only viable way to reduce the risk of impact is to reduce the time duration of the traverse. More specifically, (A.4) shows that one can optimize the value of l to reduce the time of flight of the traverse trajectory. On the other hand, (A.3b) and (A.4) show that reducing l leads to an increase in the norm of the initial velocity $\mathbf{v}_0$. Intuitively speaking, this is due to the

fact that, for a given value of d, if the time of flight decreases, the velocity along the $\mathbf{e}_2$ axis has to increase to let the vehicle cover the same distance in a shorter time. The initial velocity also depends on d, which can be tuned to reduce the velocity at the start of the traverse. The value of d cannot be chosen arbitrarily small for two reasons: (i) it is necessary to guarantee a safety margin between the quadrotor and the gap at the beginning of the traverse; (ii) the gap might not be visible during the final part of the approach trajectory. For this reason, we compute the values of the traverse trajectory parameters solving the following optimization problem:

$$\min_{d,l} t_c \quad \text{s.t.} \quad \|\mathbf{v}_0\| \leq \mathrm{v}_{0,\max}, \quad d \geq d_{\min}, \tag{A.6}$$

where $\mathrm{v}_{0,\max}$ and $d_{\min}$ are the maximum velocity allowed at the start of the traverse and the minimum value of d, respectively. We solve the nonlinear optimization problem described by (A.6) with Sequential Quadratic Programming (SQP [89], using to the NLopt library [78]. Thanks to the small dimensionality of the problem, it can be solved onboard in few tens of milliseconds.

A.2.3 Approach Trajectory

Once the traverse trajectory has been computed, its initial conditions (namely, position, velocity, and acceleration) are known. Now we can compute an approach trajectory from a suitable start position to these initial conditions. Note that this start position is not the current hover position but also results from the proposed trajectory generation method. Our goal in this step is to find a trajectory that not only matches the initial conditions of the traverse trajectory, but also enables robust perception and state estimation with respect to the gap.

Robust state estimation with respect to the gap can only be achieved by always keeping the gap in the field of view of a forward-facing camera onboard the quadrotor. Since it is difficult to incorporate these constraints into the trajectory generation directly, we first compute trajectory candidates and then evaluate their suitability for the given perception task. To do so, we use the approach proposed in [130], where a fast method to generate feasible trajectories for flying robots is presented. In that paper, the authors provide both a closed-form solution for motion primitives that minimize the jerk and a feasibility check on the collective thrust and angular velocities. The benefit of using such a method is twofold. First, it allows us to obtain a wide variety of candidate trajectories within a very short amount of time by uniformly sampling the start position and the execution time within suitable ranges. This way we can quickly evaluate a large set of candidate trajectories and select the best one according to the optimality criterion described in Sec. A.2.5. Each of these candidate trajectories consists of the quadrotor's 3D position and its derivatives. Second, and most importantly, since the computation and the verification of each trajectory takes on average a two tenths of millisecond, it is

possible to replan the approach trajectory at each control step, counteracting the effects of the uncertainty in the pose estimation of the quadrotor when it is far away from the gap. Each new approach trajectory is computed using the last state estimate available. In the following, we describe how we plan a yaw-angle trajectory for each candidate and how we select the best candidate to be executed.

A.2.4 Yaw-Angle Planning

In [110], the authors proved that the dynamic model of a quadrotor is *differentially flat*. Among other things, this means that the yaw angle of the quadrotor can be controlled independently of the position and its derivatives. In this section, we present how to compute the yaw angle such that a camera mounted on the quadrotor always faces the gap. Ideally, the camera should be oriented such that the center of the gap is projected as close as possible to the center of the image, which yields the maximum robustness for visual state estimation with respect to the gap against disturbances on the quadrotor.

To compute the desired yaw angle, we first need to compute the ideal orientation of the camera. Let $\mathbf{p}_G$ be the coordinates of the center of the gap with respect to the world frame W. Furthermore, let $\mathbf{R}_{WC}$ and $\mathbf{p}_C$ be the extrinsic parameters of the camera: $\mathbf{p}_C$ is the camera's position and the rotation matrix $\mathbf{R}_{WC} = (\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3)$ defines the camera orientation with respect to the world frame, where $\mathbf{r}_3$ is the camera's optical axis.

For a given trajectory point, we can compute the vector from the camera to the center of the gap $\mathbf{d} = \mathbf{p}_G - \mathbf{p}_C$. Ideally, we can now align the camera's optical axis $\mathbf{r}_3$ with $\mathbf{d}$ but since the trajectory constrains the quadrotor's vertical axis $\mathbf{z}_b$, we can generally not do this. Therefore, we minimize the angle between $\mathbf{d}$ and $\mathbf{r}_3$ by solving the following constrained optimization problem:

$$\mathbf{r}_3^* = \arg\max_{\mathbf{x}} \langle \mathbf{x}, \mathbf{d} \rangle \quad \text{s.t. } \|\mathbf{x}\| = 1, \quad \langle \mathbf{x}, \mathbf{z}_b \rangle = k, \tag{A.7}$$

where the last constraint says that the angle between the quadrotor's vertical body axis $\mathbf{z}_b$ and the camera's optical axis is constant and depends on how the camera is mounted on the vehicle. For example, $k = 0$ if the camera is orthogonal to the $\mathbf{z}_b$ axis as it is the case in our setup with a forward-facing camera.

Letting $\mathbf{d}_{\perp z_b} = \mathbf{d} - \langle \mathbf{d}, \mathbf{z}_b \rangle \mathbf{z}_b$ be the component of $\mathbf{d}$ perpendicular to $\mathbf{z}_b$, the solution of (A.7) is

$$\mathbf{r}_3^* = \sqrt{1 - k^2} \frac{\mathbf{d}_{\perp z_b}}{\|\mathbf{d}_{\perp z_b}\|} + k\mathbf{z}_b, \tag{A.8}$$

which is a vector lying in the plane spanned by $\mathbf{d}$ and $\mathbf{z}_b$, and the minimum angle

between the ideal and the desired optical axis is $\arccos(\langle \mathbf{r}_3^*, \mathbf{d} \rangle / \|\mathbf{d}\|)$, i.e.,

$$\theta_{min} = \arccos\Big((\sqrt{1-k^2}\|\mathbf{d}_{\perp z_b}\| + k \langle \mathbf{d}, \mathbf{z}_b \rangle) / \|\mathbf{d}\| \Big). \tag{A.9}$$

Once $\mathbf{r}_3^*$ is known, we can compute the yaw angle such that the actual camera optical axis $\mathbf{r}_3$ is aligned with $\mathbf{r}_3^*$.

Observe that in the particular case of a trajectory point that allows to align $\mathbf{r}_3$ with $\mathbf{d}$, we have $\langle \mathbf{d}, \mathbf{z}_b \rangle = k\|\mathbf{d}\|$ and the solution of (A.7) reduces to $\mathbf{r}_3^* = \frac{\mathbf{d}}{\|\mathbf{d}\|}$, with a minimum angle $\theta_{\min} = \arccos(\langle \mathbf{r}_3, \mathbf{d} \rangle / \|\mathbf{d}\|) = \arccos(1) = 0$.

A.2.5 Selection of the Approach Trajectory to Execute

In the previous sections, we described how we compute a set of candidate trajectories in 3D space and yaw for approaching the gap. All the candidate trajectories differ in their start position and their execution time. From all the computed candidates, we select the one that provides the most reliable state estimate with respect to the gap. As a quality criterion for this, we define a cost function J composed of two terms:

- the Root Mean Square (RMS) θ_{rms} of (A.9) over every sample along a candidate trajectory;
- the straight-line distance d_0 to the gap at the start of the approach.

More specifically:

$$J = \frac{\theta_{\text{rms}}}{\bar{\theta}} + \frac{d_0}{\bar{d}}, \tag{A.10}$$

where $\bar{\theta}$ and $\bar{d}$ are normalization constants that make it possible to sum up quantities with different units, and render the cost function dimensionless. This way, the quadrotor executes the candidate approach trajectory that keeps the center of the gap as close as possible to the center of the image for the entire trajectory, and at the same time prevents the vehicle from starting too far away from the gap.

A.2.6 Recovery after the Gap

Since we localize the quadrotor with respect to the gap in order to traverse it, the quadrotor is left with no state estimate after the traversal. Therefore, at this point it has to recover a vision-based state estimate and then hover in a fixed position without colliding with the environment. We solve this problem using the automatic recovery system detailed in [42], where the authors provide a method to let a quadrotor stabilize automatically after an aggressive maneuver, e.g. after a manual throw in the air.

A.3 State Estimation

A.3.1 State Estimation from Gap Detection

Our proposed trajectory generation approach is independent of the gap-detection algorithm being used; thus, to simplify the perception task, we use a black-and-white rectangular pattern to detect the gap (cf. Fig. A.1). A valid alternative to cope with real-world gaps would be to use monocular dense-reconstruction methods, such as REMODE [144]; however, they require more computing power (GPUs).

We detect the gap in each image from the forward-facing camera by applying a sequence of steps: first, we run the Canny edge detector, undistort all edges, and group close edges [180]; then, we search for quadrangular shapes and run geometrical consistency checks. Namely, we search for a quadrangle that contains another one and check the area ratio of these two quadrangles. Finally, we refine the locations of the eight corners to sub-pixel accuracy using line intersection.

Since the metric size of the gap is known, we estimate the 6-DOF pose by solving a Perspective-n-Points (PnP) problem (where $n = 8$ in our case). As a verification step, we require that the reprojection error is small. We then refine the pose by minimizing also the reprojection error of all edge pixels. To speed up the computation, we only search the gap in a region of interest around the last detection. Only when no detection is found, the entire image is searched. The detector runs with a frequency of more than 30 Hz onboard the quadrotor.

Finally, we fuse the obtained pose with IMU measurements to provide a full state estimate using the multi-sensor fusion framework of [103].

A.4 Experiments

A.4.1 Experimental Setup

We tested the proposed framework on a custom-made quadrotor, assembled from off-the-shelf hardware, 3D printed parts, and self-designed electronic components (see Fig. A.4). The frame of the vehicle is composed of a 3D printed center cross and four carbon fiber profiles as arms. Actuation is guaranteed by four RCTimer MT2830 motors, controlled by Afro Slim ESC speed controllers. The motors are tilted by 15° to provide three times more yaw-control action, while only losing 3 % of the collective thrust.

Our quadrotor is equipped with a PX4FMU autopilot that contains an IMU and a micro controller on which our custom low-level controller runs. Trajectory planning, state estimation and high-level control run on an Odroid-XU4 single-board computer. Our algorithms have been implemented in ROS, running on Ubuntu 14.04. Communication

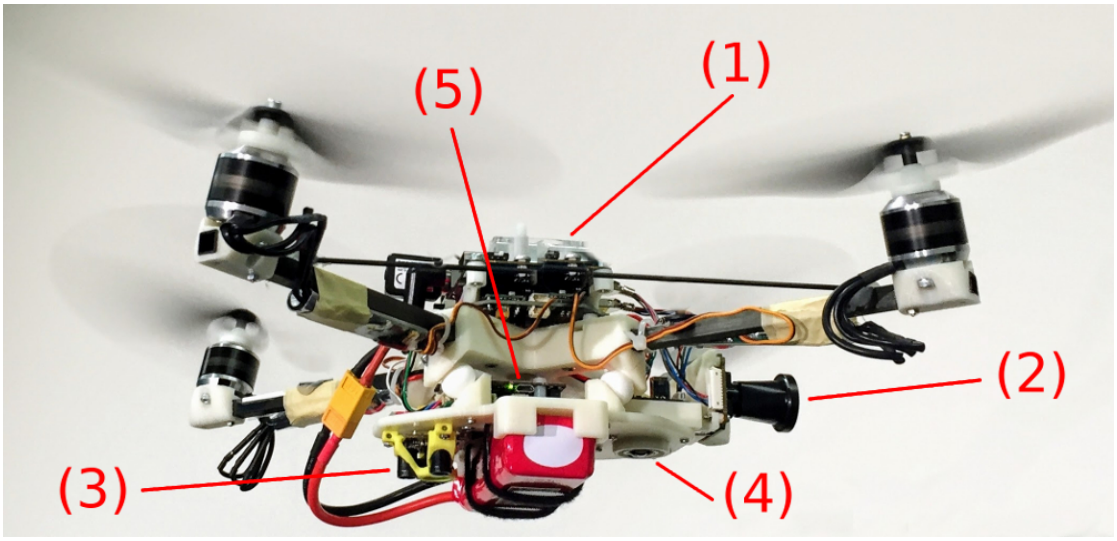


Figure A.4: The quadrotor platform used in the experiments. (1) Onboard computer. (2) Forward-facing fisheye camera. (3) TeraRanger One distance sensor and (4) downward-facing camera, both used *solely during the recovery phase*. (5) PX4 autopilot. The motors are tilted by 15° to provide three times more yaw-control action, while only losing 3 % of the collective thrust.

between the Odroid and the PX4 runs over *UART*.

Gap-detection is done through a forward-facing fisheye camera (MatrixVision mvBlueFOX-MLC200w 752 × 480-pixel monochrome camera with a 180° lens), which ensures that the gap can be tracked until very close. To allow the robot to execute the recovery maneuver after traversing the gap, we mounted the same hardware detailed in [42], which consists of a TeraRanger One distance sensor and a downward-facing camera. Notice, however, that these are *not used* for state estimation before passing the gap but *only* to recover and switch into stable hovering after the traverse.

The overall weight of the vehicle is 830 g, while its dimension are 55 × 12 cm (largest length measured between propeller tips). The dimensions of the rectangular gap are 80 × 28 cm. When the vehicle is at the center of the gap, the tolerances along the long side and short sides are only 12.5 cm, and 8 cm, respectively (cf. Fig. A.5). This highlights that the traverse trajectory must be followed with centimeter accuracy to avoid a collision.

The parameters of the traverse trajectory (Sec. A.2.2) have been set as $v_{0,max} = 3\,m\,s^{-1}$, $d_{min} = 0.25\,cm$. The normalization constants $\bar{\theta}$ and $\bar{d}$, introduced in Sec. A.2.5, have been manually tuned to let the quadrotor start the maneuver close enough to render vision-based pose estimation reliable and, at the same time, keep the gap as close as possible to the center of the image.

The dynamic model and the control algorithm used in this work are the same presented

Figure A.5: Our quadrotor during a traverse.

in [42]. We refer the reader to that for further details.

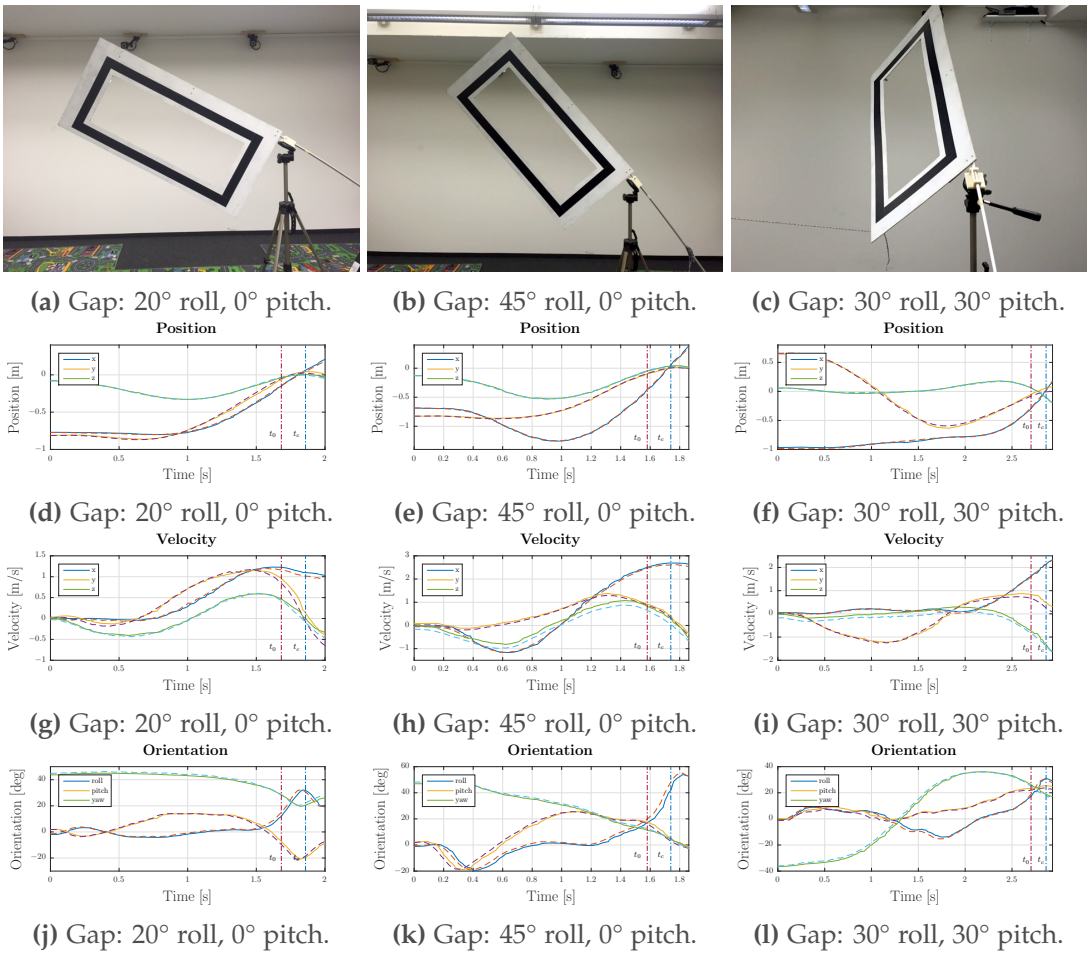
A.4.2 Results

To demonstrate the effectiveness of the proposed method, we flew our quadrotor through a gap inclined at different orientations. We consider both rotations around the world x and y axes, and denote them as *roll* and *pitch*, respectively. Overall, we ran 35 experiments with the roll angle ranging between 0° and 45° and the pitch angle between 0° and 30°. We discuss the choice of these values in Sec. A.5.3. With the gap inclined at 45°, the quadrotor reaches speeds of $3\,\mathrm{m\,s^{-1}}$ and angular velocities of $400\,^\circ\,\mathrm{s^{-1}}$.

We define an experiment as successful if the quadrotor passes through the gap without collision and recovers and locks to a hover position. We achieved a remarkable success rate of 80%. When failure occurred, we found this to be caused by a persistent absence of a pose estimate from the gap detector during the approach trajectory. This led to a large error in matching the initial conditions of the traverse trajectory, which resulted in a collision with the frame of the gap.

Figure A.6 shows the estimated position, velocity, and orientation against ground truth for some of the most significant experiments and for different orientations of the gap (namely: 20° roll, 0° pitch; 45° roll, 0° pitch; and 30° roll, 30° pitch). Ground truth is recorded from an OptiTrack motion-capture system. It can be observed that the desired trajectories were tracked remarkably well. Table A.1 reports the statistics of

the errors when the quadrotor passes through the plane in which the gap lies (i.e., at $t = t_c$), measured as the distance between actual and desired state. These statistics include both the successful and the unsuccessful experiments. The average of the norm of the position error at the center of the gap was 0.06 m, with a standard deviation of 0.05 m. The average of the norm of the velocity error was below 0.19 m s^{-1}, with a standard deviation of 0.20 m s^{-1}. We refer the reader to the attached video for further experiments with different orientations of the gap. Figure A.7 shows a picture of one of the experiments with the executed approach and traverse trajectories marked in color.



(a) Gap: 20° roll, 0° pitch. **(b)** Gap: 45° roll, 0° pitch. **(c)** Gap: 30° roll, 30° pitch.

(d) Gap: 20° roll, 0° pitch. **(e)** Gap: 45° roll, 0° pitch. **(f)** Gap: 30° roll, 30° pitch.

(g) Gap: 20° roll, 0° pitch. **(h)** Gap: 45° roll, 0° pitch. **(i)** Gap: 30° roll, 30° pitch.

(j) Gap: 20° roll, 0° pitch. **(k)** Gap: 45° roll, 0° pitch. **(l)** Gap: 30° roll, 30° pitch.

Figure A.6: Comparison between ground truth and estimated position (top), velocity (center), and orientation (bottom). Each column depicts the result of an experiment conducted with a different configuration of the gap: d, g and j 20° of roll and 0° of pitch; e, h and k 45° of roll and 0° of pitch; f, i and l 30° of roll and 30° of pitch. The approach trajectory starts at $t = 0$ and ends at $t = t_0$, when the traverse trajectory is executed. The quadrotor reaches the center of the gap at $t = t_c$ and starts the recovery maneuver at the final time of each plot. We refer the reader to the accompanying video for further experiments with different orientations of the gap.

	Position [m]			Velocity [m s^{-1}]			Orientation [°]	
	x	**y**	**z**	**x**	**y**	**z**	**roll**	**pitch**
μ	0.04	0.04	0.03	0.09	0.15	0.08	6.04	8.89
σ	0.03	0.02	0.03	0.08	0.10	0.06	3.70	5.85

Table A.1: Position, velocity and orientation error statistics at time $t = t_c$. The mean error μ and the standard deviation σ are computed using ground truth data gathered from 35 experiments conducted with the gap at different orientations.

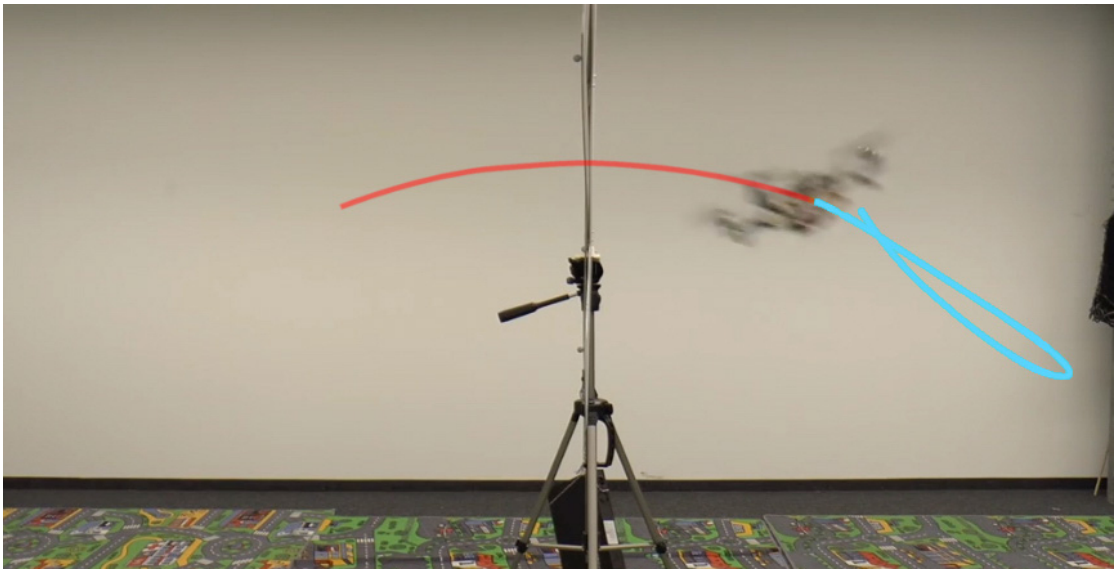


Figure A.7: Our quadrotor executing the whole trajectory split into approach (blue), traverse (red).

A.5 Discussion

In this section, we discuss our approach and provide more insights into our experiments.

A.5.1 Replanning

The method we use to compute the approach meneuver [130] can fail to verify whether a trajectory is feasible or not, as also highlighted by the authors. This usually happens when the time duration of the trajectory is short. In such a case, we skip the replanning and provide the last available approach trajectory to our controller.

A.5.2 Trajectory Computation Times

The trajectory planning approach we adopt for the approach phase is fast enough to compute and test $40,000$ trajectories in less then one second, even with the additional computational load induced by our check on the gap perception. The computation of each trajectory on the on-board computer takes on average (0.240 ± 0.106) ms, including: (i) generation of the trajectory; (ii) feasibility check; (iii) trajectory sampling and computation of the yaw angle for each sample; (iv) evaluation of the cost function described in (A.10); (v) comparison with the current best candidate. It is important to point out that these values do not apply to the *replanning* of the approach trajectory during its execution, since the initial state is constrained by the current state of the vehicle and there is no cost function to evaluate. In such a case, the computation is much faster and for each trajectory it only takes (0.018 ± 0.011) ms on average.

A.5.3 Gap configuration

Our trajectory generation formulation is able to provide feasible trajectories with any configuration of the gap, e.g., when the gap is perfectly vertical (90° roll angle) or perfectly horizontal (90° pitch angle). However, in our experiments we limit the roll angle of the gap between 0° and 45° and the pitch angle between 0° and 30°. We do this for two reasons. First, when the gap is heavily pitched, the quadrotor needs more space to reach the initial conditions of the traverse from hover. This renders the gap barely or not visible at the start of the approach, increasing the uncertainty in the pose estimation. Second, extreme configurations, such as roll angles of the gap up to 90°, require high angular velocities in order to let the quadrotor align its orientation with that of the gap. This makes gap detection difficult, if not impossible, due to motion blur. Also, our current experimental setup does not allow us to apply the torques necessary to reach high angular velocities because of the inertia of the platform and motor saturations.

A.5.4 Dealing with Missing Gap Detections

The algorithm proposed in Sec. A.3.1 fuses the poses from gap detection with IMU readings to provide the full state estimate during the approach maneuver. In case of motion blur, due to high angular velocities, or when the vehicle is too close to the gap, the gap detection algorithm does not return any pose estimate. However, these situations do not represent an issue during short periods of time (a few tenths of a second). In these cases, the state estimate from the sensor fusion module is still available and reliable through the IMU.

A.6 Conclusion

We developed a system that lets a quadrotor vehicle safely pass through a narrow inclined gap using only onboard sensing and computing. Full state estimation is provided by fusing gap detections from a forward-facing onboard camera and an IMU.

To tackle the problems arising from the varying uncertainty from the vision-based state estimation, we coupled perception and control by computing trajectories that facilitate state estimation by always keeping the gap in the image of the onboard camera.

We successfully evaluated and demonstrated the approach in many real-world experiments. To the best of our knowledge, this is the first work that addresses and achieves autonomous, aggressive flight through narrow gaps using only onboard sensing and computing, and without requiring prior knowledge of the pose of the gap. We believe that this is a major step forward autonomous quadrotor flight in complex environments with onboard sensing and computing.

B Perception-Aware Model Predictive Control

D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza. "PAMPC: Perception-Aware Model Predictive Control for Quadrotors". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Oct. 2018. DOI: 10.1109/IROS.2018.8593739

PAMPC: Perception-Aware Model Predictive Control for Quadrotors

Davide Falanga, Philipp Foehn, Peng Lu and Davide Scaramuzza

Abstract — We present the first perception-aware model predictive control framework for quadrotors that unifies control and planning with respect to action and perception objectives. Our framework leverages numerical optimization to compute trajectories that satisfy the system dynamics and require control inputs within the limits of the platform. Simultaneously, it optimizes perception objectives for robust and reliable sensing by maximizing the visibility of a point of interest and minimizing its velocity in the image plane. Considering both perception and action objectives for motion planning and control is challenging due to the possible conflicts arising from their respective requirements. For example, for a quadrotor to track a reference trajectory, it needs to rotate to align its thrust with the direction of the desired acceleration. However, the perception objective might require to minimize such rotation to maximize the visibility of a point of interest. A model-based optimization framework, able to consider both perception and action objectives and couple them through the system dynamics, is therefore necessary. Our perception-aware model predictive control framework works in a receding-horizon fashion by iteratively solving a non-linear optimization problem. It is capable of running in real-time, fully onboard our lightweight, small-scale quadrotor using a low-power ARM computer, together with a visual-inertial odometry pipeline. We validate our approach in experiments demonstrating (i) the conflict between perception and action objectives, and (ii) improved behavior in extremely challenging lighting conditions.

Supplementary material

Video: https://youtu.be/9vaj829vE18

Code: https://github.com/uzh-rpg/rpg_mpc

B.1 Introduction

Thanks to the progresses in perception algorithms, the availability of low-cost cameras, and the increased computational power of small-scale computers, vision-based perception has recently emerged as the de facto standard in onboard sensing for micro aerial vehicles. This made it possible to replicate some of the impressive quadrotor maneuvers seen in the last decade [112, 110, 130, 18], which relied on motion-capture systems, using only onboard sensing, such as cameras and IMUs [46, 99, 179].

Cameras have a number of advantages over other sensors in terms of weight, cost, size, power consumption and field of view. However, vision-based perception has severe limitations: it can be intermittent and its accuracy is strongly affected by both the environment (e.g., texture distribution, light conditions) and motion of the robot (e.g., motion blur, camera pointing direction, distance from the scene). This means that one cannot always replace motion-capture systems with onboard vision, since the motion of a camera can negatively affect the quality of the estimation, posing hard bounds on the agility of the robot. On the other hand, perception can benefit from the robot motion if it is planned considering the necessities and the limitations of onboard vision. For example, to pass through a narrow gap while localizing with respect to it using an onboard camera, it is necessary to guarantee that the gap is visible at all times. Similarly, to navigate through an unknown environment, it is necessary to guarantee that the camera always points towards texture-rich regions.

To fully leverage the agility of autonomous quadrotors, it is necessary to create synergy between perception and action by considering them jointly as a single problem.

B.1.1 Contributions

Model Predictive Control (MPC) has become increasingly popular for quadrotor control [81, 136, 9] thanks to its capability of simultaneously dealing with different constraints and objectives through optimization. In this work, we present an MPC algorithm for quadrotors able to optimize both action and perception objectives.

Our framework satisfies the robot dynamics and computes feasible trajectories with respect to the input saturations. Such trajectories are not constrained to specific time or space parametrization (e.g., polynomials in time or splines), and tightly couple

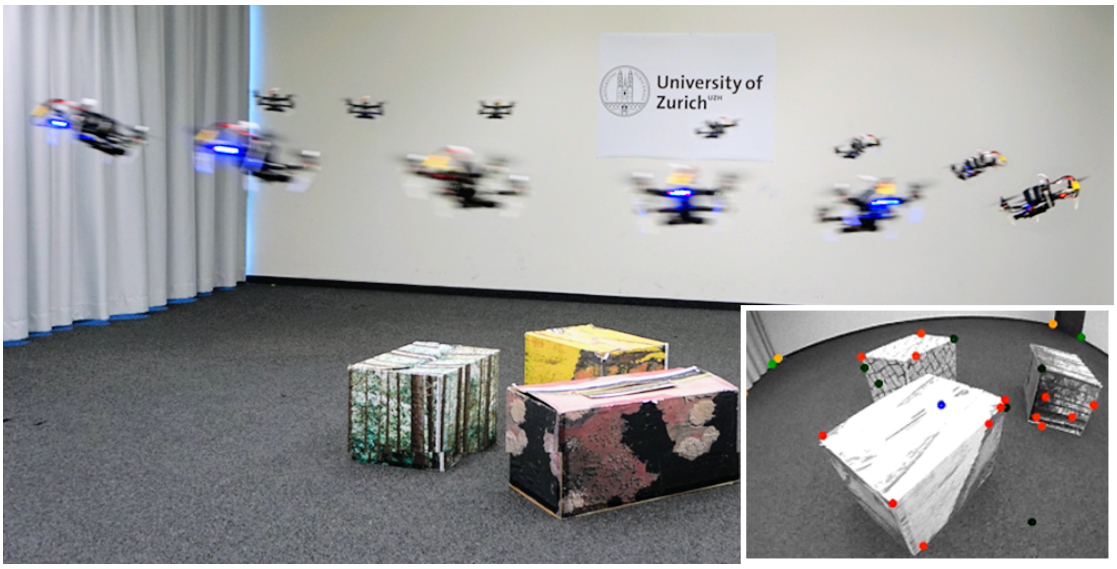


Figure B.1: An example application of our PAMPC, where a quadrotor is asked to fly at $3\,\mathrm{m\,s^{-1}}$ around a region of interest while keeping it visible in the field of view of its camera.

perception and action. To do so, perception objectives aimed at rendering vision-based estimation more robust are taken into account in the optimization problem. Such objectives are the visibility of a point of interest the robot needs to maintain in the image, and the minimization of the velocity of its projection onto the image plane. The main challenge in this is to simultaneously cope with action (e.g., dynamics, underactuation, saturations) and perception objectives, due to the potential conflicts between them.

To solve this problem, we leverage numerical optimization to compute trajectories that are optimal with respect to a cost function considering both the dynamics of the robot and the quality of perception. To fully exploit the agility of a quadrotor, we incorporate perception objectives into the optimization problem not as constraints, but rather as components to be optimized. This results in a perception-aware framework which is intrinsically tailored to agile navigation, since the optimizer can trade off between perception and action objectives (cf. Fig. B.1, depicting fast circle flight while adjusting the heading to look at a point of interest). Furthermore, considering perception in the cost function reduces the computation load of the model predictive control pipeline, allowing it to run in real-time on a low-power onboard computer. Our approach does not depend on the task and can potentially provide benefits to a large variety of applications, such as vision-based localization, target tracking, visual servoing, and obstacle detection. We validate our perception-aware model predictive control framework in real-world experiments using a small-scale, lightweight quadrotor platform.

B.1.2 Related Work

The aforementioned shift from offboard to onboard sensing based on cameras resulted in an increased number of works trying to connect perception and action.

In [141], the authors proposed a method to compute minimum-time trajectories that take into account the limited field of view of a camera to guarantee visibility of points of interests. Such a method requires the trajectory to be parametrized as a B-spline polynomial, constraining the kind of motion the robot can perform. Also, perception is included in the planning problem as hard constraint, posing an upper-bound to the agility of the robot since such constraints must be satisfied at all times. Furthermore, the velocity of the projection of the points of interest in the image is not taken into account. Finally, the algorithm was not suited for real-time control of a quadrotor, and was only tested in simulations..

In [174], the authors focused on combining visual servoing with active Structure from Motion and proposed a solution to modify the trajectory of a camera in order to increase the quality of the reconstruction. In such a work, a trajectory for the tracked features in the image plane was required, and the null space of the visual servoing task was exploited in order to render it possible for such feature to track the desired trajectory. Furthermore, the authors did not consider the underactuation of the robot, which can significantly lower the performance of the overall task due to potentially conflicting dynamics and perception objectives.

In [28] and [57], information gain was used to bridge the gap between perception and action. In the first work, the authors tackled the problem of selecting trajectories that minimize the pose uncertainty by driving the robot toward regions rich of texture. In the second work, a technique to minimize the uncertainty of a dense 3D reconstruction based on the scene appearance was proposed. In both works, however, near-hover quadrotor flight was considered, and the underactation of the platform was not taken into account.

In [169], a hybrid visual servoing technique for differentially flat systems was presented. A polynomial parameterization of the flat outputs of the system was required, and due to the computational load required by the designed optimization framework, an optimal trajectory was computed in advance and never replanned. This did not allow coping with external disturbances and unmodelled dynamics, which during the execution of the trajectory can lead to behaviours different from the expected one.

In [133] and [134], a real-time motion planning method for aerial videography was presented. In these works, the main goal was to optimize the viewpoint of a pan-tilt camera carried by an aerial robot in order to improve the quality of the video recordings. Both works were mainly targeted to cinematography, therefore they considered objectives such as the size of a target of interest and its visibility. Conversely, we target

robotic sensing and consider objectives aimed at facilitating vision-based perception.

In [145], the authors proposed a two-step approach for target-aware visual navigation. First, position-based visual servoing was exploited to find a trajectory minimizing the reprojection error of a landmark of interest. Then, a model predictive control pipeline was used to track such a trajectory. Conversely, we solve the trajectory optimization and tracking within a single framework. Additionally, that work only aimed at rendering the target visible, but did not take into account that, due to the motion of the camera, it might not be detectable because of motion blur. We cope with this problem by considering in the optimization problem the velocity of the projection of the point of interest in the image plane.

B.1.3 Structure of the Paper

The remainder of this paper is organized as follows. In Sec. B.2 we provide the general formulation of the problem. In Sec. B.3 we derive the model for the dynamics of the projection of a 3D point into the image plane for the case of a quadrotor equipped with a camera. In Sec. B.4 we present our perception-aware optimization framework, describing the objectives and the constraints it takes into account. In Sec. B.5 we validate our approach in different real-world experiments showcasing the capability of our framework. In Sec. B.6 we discuss our approach and provide additional insights and in Sec. B.7 we draw the conclusions.

B.2 Problem Formulation

For truly autonomous robot navigation, two components are essential: (I) perception, both of the ego-motion and of the surrounding environment; (II) action, meant as the combination of motion planning and control algorithms. A very wide literature is available for both of them. However, they are rarely considered as a joint problem.

The need for coupled perception and action can be easily explained. To guarantee safety, accurate and robust perception is necessary. Nevertheless, the quality of vision-based perception is strongly affected by the motion of the camera. On the one hand, it can degrade its performance by not making it possible to extract sufficiently accurate information from images. For example, lack of texture or blur due to camera motion can lead to algorithm failure. On the other, the quality of vision-based perception can improve significantly if its limitations and requirements are considered, e.g. by rendering highly-textured areas visible in the image and by reducing motion blur. Therefore it is necessary to create synergy between perception and action.

Let $\mathbf{x}$ and $\mathbf{u}$ be the state and input vectors of a robot, respectively. Assume its dynamics to be described by a set of differential equations $\dot{\mathbf{x}} = \boldsymbol{f}(\mathbf{x}, \mathbf{u})$. Furthermore, let $\mathbf{z}$ be the

state vector of the perception system (e.g., 3D points' projection onto the image plane), and $\boldsymbol{\alpha}$ a vector of parameters characterizing it (e.g., the focal length of the camera or its field of view). The perception state and the robot state are coupled through the robot dynamics, namely $\mathbf{z} = \boldsymbol{f}_p(\mathbf{x}, \mathbf{u}, \boldsymbol{\alpha})$. Given certain action objectives, we can define an action cost $\mathcal{L}_a(\mathbf{x}, \mathbf{u})$. Similarly, we can define a cost $\mathcal{L}_p(\mathbf{z})$ for the perception objectives.

We can then formulate the coupling of perception and action as an optimization problem:

$$\begin{aligned} \min_{\mathbf{u}} \quad & \int_{t_0}^{t_f} \mathcal{L}_a(\mathbf{x}, \mathbf{u}) + \mathcal{L}_p(\mathbf{z})\, dt \\ & \text{subject to} \quad \mathbf{r}(\mathbf{x}, \mathbf{u}, \mathbf{z}) = 0 \\ & \qquad\qquad\quad \mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{z}) \leq 0, \end{aligned} \tag{B.1}$$

where $\mathbf{r}(\mathbf{x}, \mathbf{u}, \mathbf{z})$ and $\mathbf{h}(\mathbf{x}, \mathbf{u}, \mathbf{z})$ represent equality and inequality constraints that the solution should satisfy for perception, action, or both of them simultaneously.

B.3 Methodology

Any computer vision algorithm aimed at providing a robot with the information necessary for navigation (e.g., pose estimation, obstacle detection, etc) has two fundamental requirements. First, the points of interest used by the algorithm to provide the aforementioned information must be visible in the image. For example, such points can be the landmarks used for pose estimation by visual odometry algorithms, or the points belonging to an object for obstacle detection. If such points are not visible while the robot is moving, there is no way the algorithm can cope with the absence of information. Second, such points of interest must be clearly recognizable in the image. Depending on the motion of the camera and the distance from the scene, the projection of a 3D point onto an image can suffer from motion blur, making it very complicated, if not impossible, to extract meaningful information. Therefore, the motion of the camera should be thoroughly planned to guarantee robust visual perception.

Based on the considerations above, in this work we consider two perception objectives in our framework: (I) visibility of points of interest, and (II) minimization of the velocity of their projection onto the image plane. In the following, we study the relation between the motion of a quadrotor equipped with an onboard camera and the projection onto the image plane of a point in space. Without loss of generality, we consider the case of a single 3D point of interest. Our goal is to couple perception and action into an optimization framework by expressing the dynamics of its projection onto the image plane as a function of the state and input vectors of a quadrotor.

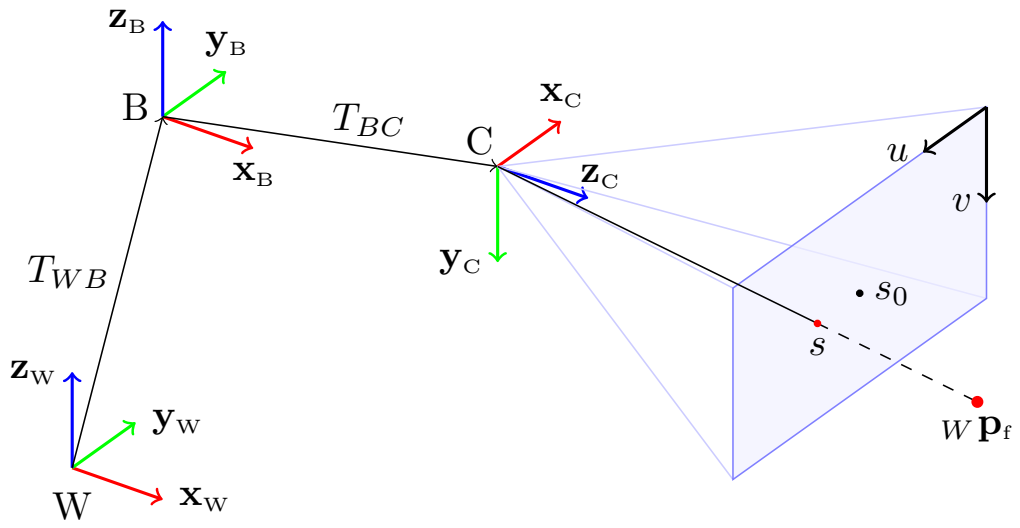


Figure B.2: A schematics representing the world frame W, the body frame B and the camera frame C. The position and orientation of B with respect to W is provided by T_{WB}. The constant rigid body transformation T_{BC} provides the extrinsics of the camera. A feature located at ${}_W\mathbf{p}_f$ is projected into the image plane onto a point of coordinates $\mathbf{s}$. $\mathbf{s}_0$ represents the principal point.

B.3.1 Nomenclature

In this work, we make use of a world frame W with orthonormal basis $\{\mathbf{x}_W, \mathbf{y}_W, \mathbf{z}_W\}$. The quadrotor frame B, also referred to as the body frame, has orthonormal basis $\{\mathbf{x}_B, \mathbf{y}_B, \mathbf{z}_B\}$. Finally, we assume the robot to be equipped with a camera, whose reference frame C has orthonormal basis $\{\mathbf{x}_C, \mathbf{y}_C, \mathbf{z}_C\}$. Fig. B.2 provides a clear overview about the reference frames.

Throughout this manuscript, we represent vectors as bold quantities having a prefix, representing the frame in which they are expressed, and a suffix, indicating the origin and the end of such a vector. For example, the quantity ${}_W\mathbf{p}_{WB}$ represents the position of the body frame B with respect to the world frame W, expressed in the world frame. To simplify the notation, if a vector has no prefix, we assume it to be expressed in the first frame reported in the suffix (i.e., the frame where the vectors origin is).

We use quaternions to represent the orientation of a rigid body. The time derivative of a quaternion $\mathbf{q} = (q_w, q_x, q_y, q_z)^\mathsf{T}$ is given by $\dot{\mathbf{q}} = \frac{1}{2}\Lambda(\boldsymbol{!}) \cdot \mathbf{q}$, where the skew-symmetric matrix $\Lambda(\boldsymbol{!})$ of a vector $\boldsymbol{!} = (\omega_x, \omega_y, \omega_z)^\mathsf{T}$ is defined as:

$$\Lambda(\omega) = \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix}. \tag{B.2}$$

Finally, we use the operator $\odot$ to denote the multiplication between a quaternion and a vector. More specifically, multiplying a vector $\mathbf{v}$ with the quaternion $\mathbf{q}$ means rotating $\mathbf{v}$ by the rotation induced by $\mathbf{q}$. By doing so, we obtain a vector $\mathbf{v}' = \mathbf{v} \odot \mathbf{q} = Q\mathbf{v}$ where:

$$Q = \begin{bmatrix} 1 - 2q_y^2 - 2q_z^2 & 2(q_x q_y + q_w q_z) & 2(q_x q_z - q_w q_y) \\ 2(q_x q_y - q_w q_z) & 1 - 2q_x^2 - 2q_z^2 & 2(q_y q_z + q_w q_x) \\ 2(q_x q_z + q_w q_y) & 2(q_y q_z - q_w q_x) & 1 - 2q_x^2 - 2q_y^2 \end{bmatrix}.$$

B.3.2 Quadrotor Dynamics

Let $\mathbf{p}_{WB} = (p_x, p_y, p_z)^\intercal$ and $\mathbf{q}_{WB} = (q_w, q_x, q_y, q_z)^\intercal$ be the position and the orientation of the body frame with respect to the world frame W, expressed in world frame, respectively (cf. Fig. B.2). Additionally, let $\mathbf{v}_{WB} = (v_x, v_y, v_z)^\intercal$ be the linear velocity of the body, expressed in world frame, and $\mathbf{\Omega}_B = (\omega_x, \omega_y, \omega_z)^\intercal$ its angular velocity, expressed in the body frame. Finally, let $\mathbf{c} = (0, 0, c)^\intercal$ be the mass-normalized thrust vector, where $c = (f_1 + f_2 + f_3 + f_4)/m$, f_i is the thrust produced by the i-th motor, and m is the mass of the vehicle. In this work, we use the dynamical model of a quadrotor proposed in [130]:

$$\begin{aligned} \dot{\mathbf{p}}_{WB} &= \mathbf{v}_{WB} \\ \dot{\mathbf{v}}_{WB} &= g + \mathbf{q}_{WB} \odot \mathbf{c} \\ \dot{\mathbf{q}}_{WB} &= \frac{1}{2}\Lambda(\mathbf{\Omega}_B) \cdot \mathbf{q}_{WB} \end{aligned} \tag{B.3}$$

where $g = (0, 0, -g)^\intercal$ is the gravity vector, with $g = 9.81\,\mathrm{m\,s^{-2}}$. The state and the input vectors of the system are $\mathbf{x} = [\mathbf{p}_{WB}, \mathbf{v}_{WB}, \mathbf{q}_{WB}]^\intercal$ and $\mathbf{u} = [c, \mathbf{\Omega}_B^\intercal]^\intercal$, respectively.

B.3.3 Perception Objectives

Let ${}_W\mathbf{p}_f = ({}_W p_{fx}, {}_W p_{fy}, {}_W p_{fz})$ be the 3D position of a point of interest (landmark) in the world frame W (cf. Fig. B.2). We assume the body to be equipped with a camera having extrinsic parameters described by a constant rigid body transformation $T_{BC} = [\mathbf{p}_{BC}, \mathbf{q}_{BC}]$, where $\mathbf{p}_{BC}$ and $\mathbf{q}_{BC}$ are the position and the orientation of C with respect to B. The coordinates ${}_C\mathbf{p}_f = ({}_C p_{fx}, {}_C p_{fy}, {}_C p_{fz})^\intercal$ of ${}_W\mathbf{p}_f$ in the camera frame C are given by:

$${}_C\mathbf{p}_f = (\mathbf{q}_{WB}\ \mathbf{q}_{BC})^{-1} \odot \left({}_W\mathbf{p}_f - (\mathbf{q}_{WB} \odot\ \mathbf{p}_{BC} + \mathbf{p}_{WB})\right). \tag{B.4}$$

The point ${}_C\mathbf{p}_f$ in camera frame is projected into the image plane coordinates $\mathbf{s} = (u, v)^\intercal$ according to classical pinhole camera model [181]:

$$u = f_x \frac{{}_C p_{fx}}{{}_C p_{fz}}, \quad v = f_y \frac{{}_C p_{fy}}{{}_C p_{fz}} \tag{B.5}$$

where f_x, f_y are the focal lengths for pixel rows and columns, respectively.

To guarantee robust vision-based perception, the projection $\mathbf{s}$ of a point of interest ${}_W\mathbf{p}_f$ should be as close as possible to the center of the image for two reasons. First, keeping its projection in the center of the image results in the highest safety margins against external disturbances. The second reason comes from the fact that the periphery of the image is typically characterized by a non-negligible distortion, especially for large field of view cameras. A number of models for such distortion are available in the literature, as well as techniques to estimate their parameters to compensate the effects of the distortion. However, such a compensation is never perfect and this can degrade the accuracy of the estimates.

As previously mentioned, in addition to rendering the point of interest visible in the image, we are interested in reducing the velocity of its projection onto the image plane. We assume the point of interest to be static, but similar considerations apply to the case where such a point of interest moves with respect to the world frame. To express the projection velocity as a function of the quadrotor state and input vectors, we can differentiate (B.5) with respect to time:

$$\begin{aligned} \dot{u} &= f_x \frac{{}_C\dot{p}_{fx}\ {}_Cp_{fz} - {}_Cp_{fx}\ {}_C\dot{p}_{fz}}{{}_Cp_{fz}^2}, \\ \dot{v} &= f_y \frac{{}_C\dot{p}_{fy}\ {}_Cp_{fz} - {}_Cp_{fy}\ {}_C\dot{p}_{fz}}{{}_Cp_{fz}^2}. \end{aligned} \tag{B.6}$$

Eq. (B.6) can be written in a compact form as:

$$\dot{\mathbf{s}} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ 0 \end{bmatrix} = \begin{bmatrix} 0 & -\frac{f_x}{{}_Cp_{fz}^2} & 0 \\ \frac{f_y}{{}_Cp_{fz}^2} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \left({}_C\mathbf{p}_f \times_C {}_C\dot{\mathbf{p}}_f \right). \tag{B.7}$$

To compute the term ${}_C\dot{\mathbf{p}}_f$, we can differentiate (B.4) with respect to time:

$${}_C\dot{\mathbf{p}}_f = -\frac{1}{2}\Lambda\left(\boldsymbol{\Omega}_C\right)\ {}_C\mathbf{p}_f - {}_C\mathbf{v}_{WC}, \tag{B.8}$$

where:

$$\begin{aligned} {}_C\mathbf{v}_{WC} &= \left(\mathbf{q}_{WB}\ \mathbf{q}_{BC}\right)^{-1} \odot \left(\frac{1}{2}\Lambda\left(\boldsymbol{\Omega}_B\right) \mathbf{q}_{WB} \odot \mathbf{p}_{BC} + \mathbf{v}_{WB} \right), \\ \boldsymbol{\Omega}_C &= \mathbf{q}_{BC}^{-1} \odot \boldsymbol{\Omega}_B. \end{aligned} \tag{B.9}$$

B.3.4 Action Objectives

For a quadrotor to execute a desired task (e.g., reach a target position in space), a suitable trajectory has to be planned. In this regard, for a quadrotor two objectives should be considered.

The first comes from the bounded inputs available to the system. The thrust each motor can produce has both an upper and a lower bound, leading to a limited input vector $\mathbf{u}$. Therefore, denoting the subset of the allowed inputs as $\mathcal{U}$, the planned trajectory should be such that the condition $\mathbf{u}(t) \in \mathcal{U}\ \forall t$ can be satisfied.

The second objective to be considered comes from the underactuated nature of a quadrotor. In the most common configuration, all the rotors point in the same direction, typically along the axis $\mathbf{z}_B$ of the body. This means that the robot can accelerate only in this direction. Therefore, to move in the 3D space, it is necessary to exploit the system dynamics (B.3) by coupling the translational and the rotational motions of the robot to follow the desired trajectory.

B.3.5 Challenges

The perception (Sec. B.3.3) and the action (Sec. B.3.4) objectives previously described are both necessary for vision-based quadrotor navigation. Considering them simultaneously is challenging due to the possible conflict among them. Indeed, for a quadrotor to track a reference trajectory, it needs to rotate to align its thrust with the direction of the desired acceleration. However, the perception objective might require to minimize such rotation to maximize the visibility of a point of interest. A model-based optimization framework able to consider both perception and action objectives and couple them through the system dynamics is therefore necessary.

B.4 Model Predictive Control

Formulating coupled perception and action as an optimization problem has the advantages of being able to satisfy the underactuated system dynamics and actuator constraints (i.e., input boundaries) and to minimize the predicted costs along a time horizon. In contrast, classical control schemes are incapable of predicting costs and the corresponding trajectory (e.g., PID controllers) and guaranteeing input boundaries (PID, LQR).

The basic formulation of such an optimization is given in (B.1), which in our case results in a non-linear program with quadratic costs. This can then be approximated by a sequential quadratic program (SQP) where the solution of the non-linear program is iteratively approximated and used as a model predictive control (MPC). To this regard,

for the MPC to be effective, the optimization scheme has to run in real-time, at the desired control frequency. To achieve this, we first discretize the system dynamics with a time step dt for a time horizon t_h into $\mathbf{x}_i \ \forall i \in [1, N]$ and $\mathbf{u}_i \ \forall i \in [1, N-1]$. We define the time-varying state cost matrix as $\mathcal{Q}_{x,i} \ \forall i \in [1, N]$. Furthermore, the time-varying perception and input cost matrices are defined as $\mathcal{Q}_{p,i}$ and $\mathcal{R}_i$, $\forall i \in [1, N-1]$, respectively. Finally, let $\mathbf{z} = [\mathbf{s}, \dot{\mathbf{s}}]$ be the perception function. It is important to recall that $\mathbf{z}$ is a function of the quadrotor's state and input variables, as remarked in Eq. (B.4) to (B.9). The resulting cost function we consider is:

$$\mathcal{L} = \bar{\mathbf{x}}_N^\intercal \mathcal{Q}_{x,N} \bar{\mathbf{x}}_N + \sum_{i=1}^{N-1} \left(\bar{\mathbf{x}}_i^\intercal \mathcal{Q}_{x,i} \bar{\mathbf{x}}_i + \bar{\mathbf{z}}_i^\intercal \mathcal{Q}_{p,i} \bar{\mathbf{z}}_i + \bar{\mathbf{u}}_i^\intercal \mathcal{R}_i \bar{\mathbf{u}}_i \right), \tag{B.10}$$

where the values $\bar{\mathbf{x}}, \bar{\mathbf{z}}, \bar{\mathbf{u}}$ refer to the difference with respect to the reference of each value. In our case, the reference value for $\mathbf{z}$ is the null vector (i.e., center of the image and zero velocity) and the reference for the states and inputs are given by a target pose or a precomputed trajectory (that neglects the perception objectives).

The inputs $\mathbf{u}$, consisting of c and $\boldsymbol{\Omega}_B$, as well as the velocity $\mathbf{v}_{WB}$ are limited by the constraints:

$$c_{min} \leq c \leq c_{max}, \tag{B.11}$$

$$-\Omega_{max} \leq \boldsymbol{\Omega}_B \leq \Omega_{max}, \tag{B.12}$$

$$-v_{max} \leq \mathbf{v}_{WB} \leq v_{max}, \tag{B.13}$$

where $c_{min}, c_{max}, \Omega_{max}, v_{max} \in \mathcal{R}_+$.

To include the dynamics as in (B.3), we use multiple shooting as transcription method and a Runge-Kutta integration scheme. We refer the reader to [76] and [75] for more details on the transcription of the dynamics for optimization.

We approximate the solution of the optimization problem by executing one iteration at each control loop and use as initial state the most recent available estimate $\mathbf{x}_{est}$ provided by a Visual-Inertial Odometry pipeline running onboard the vehicle (see Sec. B.5.1). To achieve good approximations, it is important to run these iterations significantly faster than the discretization time of the problem and to keep the previous solution as initialization trajectory of the next optimization. Such a SQP scheme leads to a fast convergence towards the exact solution, since the system is always close to the last linearization, and the deviation of each state $\mathbf{x}_i$ between two iterations is very small.

Figure B.3: The quadrotor used for the experiments.

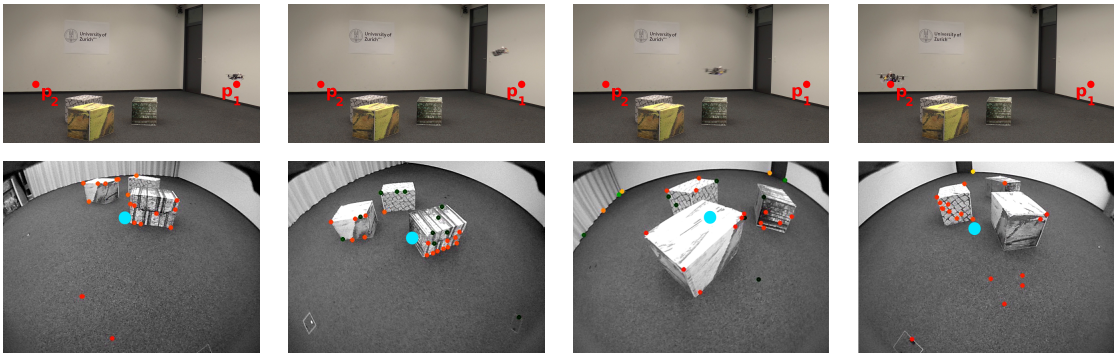
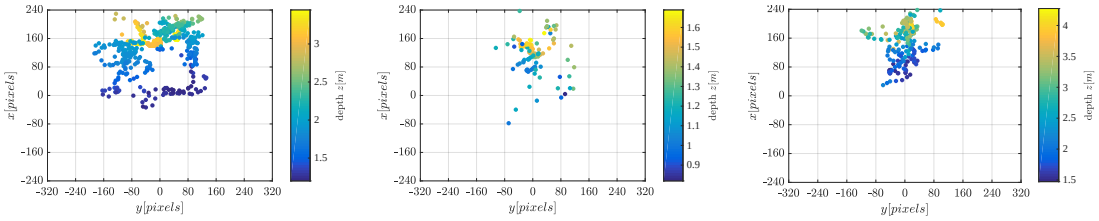


Figure B.4: A sequence of the visibility experiment for the hover-to-hover flight experiment, with time progressing from left to right. The quadrotor performs a maneuver to fly to a new reference pose, exploiting additional height to pitch less and keep the point of interest (centroid of the vision features, marked as cyan circle) in the center of the image. The corresponding footage is available in the accompanying video.

B.5 Experiments

In order to show the potential of our perception-aware model predictive control, we ran our approach onboard a small, vision-based, autonomous quadrotor. We refer the reader to the attached video showcasing the experiments.



(a) Reprojection for the circular trajectory.

(b) Reprojection for the *hover-to-hover* experiment.

(c) Reprojection for the darkness experiment.

Figure B.5: Reprojection of the point of interest in image plane, colored according to the depth with respect to the camera frame.

B.5.1 Experimental Setup

We used a small and lightweight quadrotor platform to achieve high agility through high torque-to-inertia and thrust-to-weight ratios, and improve simplicity and safety for the user (cf. Fig B.3). The quadrotor had a take-off weight of 420 g, a thrust-to-weight ratio of ~ 2, and a motor-to-motor diagonal of 220 mm. We used a Qualcomm Snapdragon Flight board with a quad-core ARM processor at up to 2.26 GHz and 2 GB of RAM, paired with a Qualcomm Snapdragon Flight ESC. The board was equipped with an Inertial Measurement Unit and a forward-looking, wide field-of-view global-shutter camera tilted down by 45° for visual-inertial odometry (VIO) using the Qualcomm mvSDK. It ran ROS on Linux and our self-developed flight stack. We setup the optimization with ACADO and used qpOASES as solver. As discretization step, we chose $dt = 0.1$ s with a time horizon of $t_h = 2$ s and ran one iteration step in each control loop with a frequency of 100 Hz. Therefore, the iteration ran roughly $10\times$ faster than the discretization time, resulting in small deviations of the predicted state vector between iterations and facilitating convergence. The code developed in this work is publicly available as open-source software.

B.5.2 Experiment Description and Results

To prove the functionality and importance of our PAMPC, we ran three experiments. In the first experiment, the controller modified a circular trajectory to improve the visibility of a point of interest. In the second experiment, the controller handled *hover-to-hover* flight by deviating from a straight line trajectory to keep the point of interest visible. In the third experiment, it enabled vision-based flight in an extremely challenging scenario. All the experiments were conducted with onboard VIO and onboard computation of the PAMPC, without any offline computation and without any motion-capture system.

Circular Flight

We setup a small pile of boxes in the middle of a room otherwise poor of texture. We did this to force the VIO pipeline to use such boxes as features for state estimation. The centroid of these features was set as our point of interest. We provided the robot with a circular reference trajectory around the aforementioned boxes and asked it to fly along such a trajectory while maintaining the boxes visible in the center of the image (cf. Fig. B.1). We evaluated the performance of our framework for speeds along the circle from $1\,\mathrm{m\,s^{-1}}$ to $3\,\mathrm{m\,s^{-1}}$.

The results of one run of the circular flight experiments at $3\,\mathrm{m\,s^{-1}}$ are depicted in Fig. B.6. Despite the agility of the maneuver, which requires large deviations from the hover conditions, the robot is able to keep the point of interest visible in the onboard image. Fig. B.5a reports the reprojection in the image plane for such point of interest.

Hover-To-Hover Flight

In this experiment within the same scenario as in Sec. B.5.2, we showed the capabilities of our framework for *hover-to-hover* flight. More specifically, we requested a pose jump from a position $\mathbf{p}_1$ to $\mathbf{p}_2$ at equal height (cf. Fig. B.4). During that maneuver, the quadrotor had to pitch down to reach the desired acceleration, but controversially should pitch as little as possible to keep the point of interest visible. A sequence of this experiment is visible in Fig. B.4.

One can easily see that, despite the start and end positions are at the same height, the quadrotor not only pitches to go towards the new reference in an horizontal motion, but also accelerates upward (i.e., in positive z, cf. Fig. B.7). This results in a smaller pitch angle and a higher thrust to reach the same y-acceleration, which is helpful for perception since it brings the features towards the center of the frame due to the higher altitude. If perception objectives were not considered, the resulting trajectory would have not required any height change, potentially leading to a poor visibility of the point of interest. The full motion of the quadrotor is depicted in Fig. B.7, where the exploitation of the added height and the orientation of the camera frame can be seen. Finally, in Fig. B.5b we show the reprojection in the image plane for the point of interest.

Darkness Scenario

This experiment was targeted towards extremely challenging scenarios, such as flight in a very dark environment, or otherwise difficult illumination conditions (cf. Fig. B.8). To demonstrate the performance in such a scenario, we flew the vehicle several times in a dark room with two illuminated spots. If the illuminated spots left the field of view for a moment, the VIO pipeline would drift quickly or even completely loose

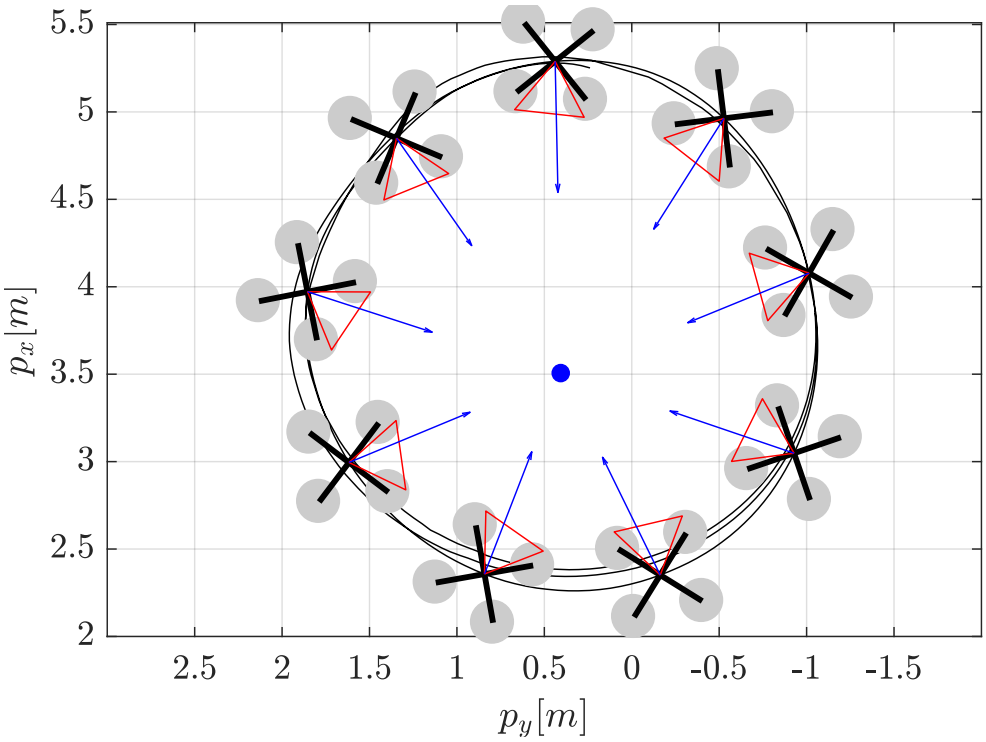


Figure B.6: Executed trajectory with quadrotor heading while the arrow points toward the point of interest (blue).

track, potentially leading to a crash. Therefore, in such scenarios it was of immense importance to keep the few available features always visible. The flown path was given by a trajectory passing through four waypoints forming a rectangle, but without any heading reference. The quadrotor correctly adjusts its heading to keep the illuminated spot in its field of view, because this is the only source of trackable features. Fig. B.9 visualizes a setup with two spotlights and a cardboard wall in between, where the quadrotor first focuses on the upper right illuminated spot, and further down the track switches to the second illuminated spot behind the wall. The reprojection of the point of interest in the image plane is shown in Fig. B.5c.

B.6 Discussion

B.6.1 Choice of the optimizer

To implement the optimization problem, we chose to use ACADO because of two main reasons: (I) it is capable of transcribing system dynamics with single- and multiple-shooting and integration schemes, as well as provide an interface to a solver; (II) it generates c++ code, which then is compiled directly on the executing platform, which allows it to use accelerators and optimizations tailored to the platform.

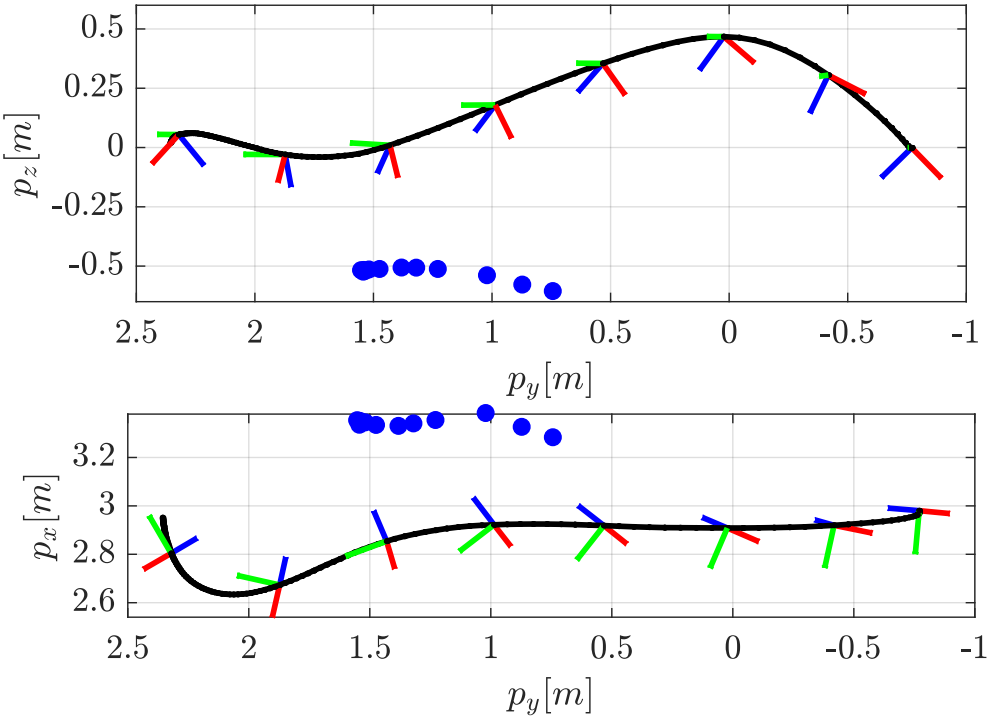


Figure B.7: Quadrotor path in hover-to-hover, looking towards the centroid of tracked features (blue), with the camera frame indicated by $\{x_C, y_C, z_C\}$.

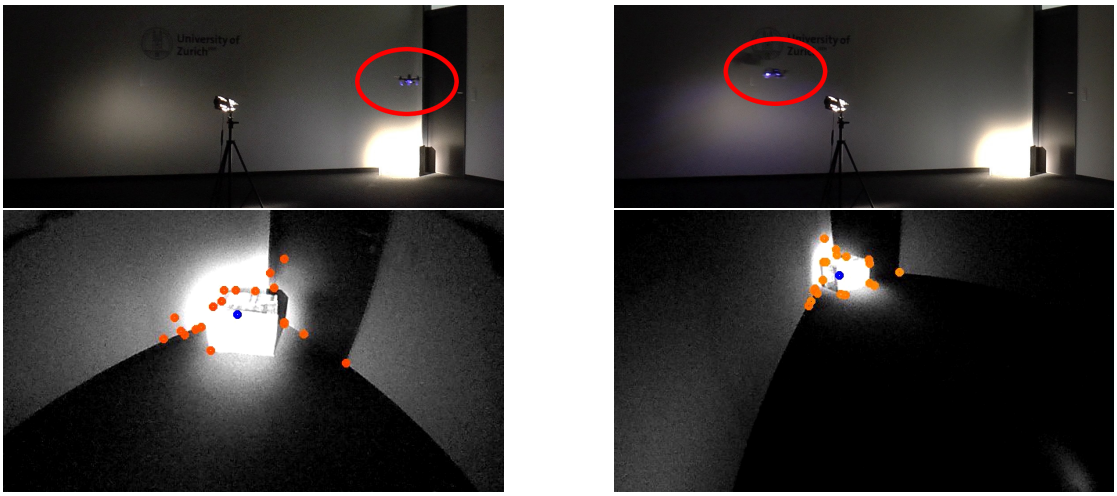


Figure B.8: A sequence of the darkness experiment with time progressing from left to right. The quadrotor, highlighted by a red circle in the figures in the first row, tracks a trajectory and adjusts its heading to keep the point of interest (centroid of the vision features) in field of view.

B.6.2 Convexity of the problem

Our state and input space is a convex domain, hence also any quadratic cost in those is convex. The perception costs could be argued to be non-convex due to the division

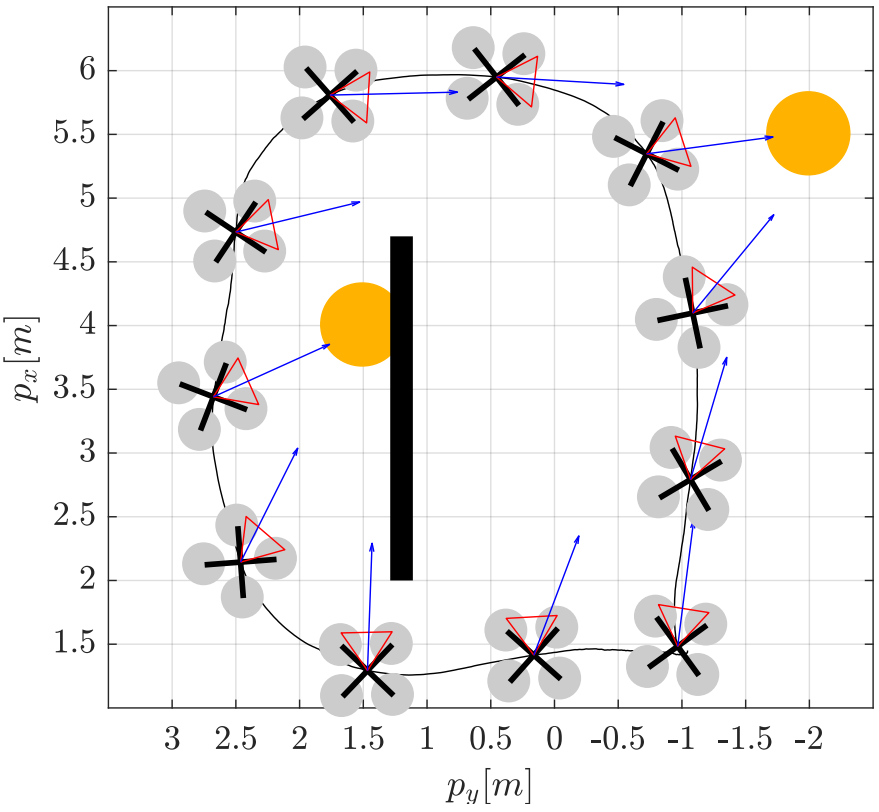


Figure B.9: Path of the quadrotor, looking towards light spots (yellow), with camera direction (red triangle) and point-of-interest direction (blue arrow).

by ${}_C p_{fz}$ in the projection (B.5). However, on examination of the projection one will notice that the denominator ${}_C p_{fz}$ is always positive, since the pinhole camera projection model does not allow negative or zero depths. We can therefore constrain ${}_C p_{fz}$ to be positive, rendering all possible solutions in the positive halfplane $\mathcal{R}^+$ and therefore recover convexity.

B.6.3 Choice of point of interest

In our experiments, we used the centroid of detected features as our point of interest. Assuming that all the features are equally important, instead of optimizing for each individually, we can summarize them as their centroid, which results in the same optimal solution.

B.6.4 PAMPC Parameters

We chose a discretization of $dt = 0.1\,\mathrm{s}$ and a time horizon of $t_h = 2\,\mathrm{s}$. One could always argue that a longer time horizon and a shorter discretization step are beneficial,

but they also increase the computation time by roughly $\mathcal{O}(N^2)$ with the number of discretization nodes $N = \frac{t_h}{dt}$. In our experience, we could not identify any significant gain from smaller discretization steps nor from a longer time horizon.

B.6.5 Computation Time

Since the computation time must be low enough to execute the optimization in a real time scheme, we show that it is significantly lower than the one required by the controller frequency of 100 Hz. Indeed, our PAMPC requires on average 3.53 ms. It is interesting to note that this is the case for both an idle CPU and while running the full pipeline with VIO and our full control pipeline. This is due to the quad-core ARM CPU and the fact that our full pipeline without the PAMPC takes up only 3 cores leaving one free for the PAMPC. However, the standard deviation increases significantly if the CPU is under load (from 0.155 ms to 0.354 ms), even though the maximal execution time always stays below 5 ms.

B.6.6 Drawbacks of a Two-Step Approach

An alternative approach to the problem tackled in this work is to use the differential flatness as in [110] to plan a translational trajectory connecting the start and end positions, and subsequentially plan the yaw angle to point the camera towards the point of interest. After planning, a suitable controller could be used to track the desired reference trajectory. Although possible, such a solution would lead to sub-optimal results because of the following reasons: (I) the roll and pitch angles of the quadrotor would be planned without considering the visibility objective, therefore might render the point of interest not visible in the image despite the yaw control; (II) because of the split between planning and control, even if the first would provide guarantees about visibility, these could not be preserved during the control stage due to deviations from the nominal trajectory; (III) it would be challenging to provide guarantees about the respect of the input saturations. Therefore, our proposed approach considering perception, planning and control as a single problem leads to superior results.

B.7 Conclusions

In this work, we presented a perception-aware model predictive control (PAMPC) algorithm for quadrotors able to optimize both action and perception objectives. Our framework computes trajectories that satisfy the system dynamics and inputs limits of the platform. Additionally, it optimizes perception objectives by maximizing the visibility of a point of interest in the image and minimizing the velocity of its projection into the image plane for robust and reliable sensing. To fully exploit the agility of a quadrotor, we incorporated perception objectives into the optimization problem not

as constraints, but rather as components in the cost function to be optimized. Our algorithm is able to run in real-time on an onboard ARM processor, in parallel with a VIO pipeline, and is used to directly control the robot. We validated our approach in real-world experiments using a small-scale, lightweight quadrotor platform.

C The Role of Perception Latency in Obstacle Avoidance

D. Falanga, S. Kim, and D. Scaramuzza. "How Fast is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid". In: *IEEE Robot. Autom. Lett.* 4.2 (Apr. 2019), pp. 1884–1891. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2898117

How Fast is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid

Davide Falanga, Suseong Kim, and Davide Scaramuzza

Abstract — In this work, we study the effects that perception latency has on the maximum speed a robot can reach to safely navigate through an unknown cluttered environment. We provide a general analysis that can serve as a baseline for future quantitative reasoning for design trade-offs in autonomous robot navigation. We consider the case where the robot is modeled as a linear second-order system with bounded input and navigates through static obstacles. Also, we focus on a scenario where the robot wants to reach a target destination in as little time as possible, and therefore cannot change its longitudinal velocity to avoid obstacles. We show how the maximum latency that the robot can tolerate to guarantee safety is related to the desired speed, the range of its sensing pipeline, and the actuation limitations of the platform (i.e., the maximum acceleration it can produce). As a particular case study, we compare monocular and stereo frame-based cameras against novel, low-latency sensors, such as event cameras, in the case of quadrotor flight. To validate our analysis, we conduct experiments on a quadrotor platform equipped with an event camera to detect and avoid obstacles thrown towards the robot. To the best of our knowledge, this is the first theoretical work in which perception and actuation limitations are jointly considered to study the performance of a robotic platform in high-speed navigation.

Supplementary material

All the videos of the experiments are available at:
http://youtu.be/sbJAi6SXOQw

C.1 Introduction

High-speed robot navigation in cluttered, unknown environments is currently an active research area [83, 22, 151, 120, 152, 10, 79] and benefits of over 50 million US dollar funding available through the DARPA Fast Lightweight Autonomy Program (2015-2018) and the DARPA Subterranean Challenge (2018-2021).

To prevent a collision with an obstacle or an incoming object, a robot needs to detect them as fast as possible and execute a safe maneuver to avoid them. The higher the relative speed between the robot and the object, the more critical the role of *perception latency* becomes.

Perception latency is the time necessary to *perceive* the environment and *process* the captured data to generate control commands. Depending on the task, the processing algorithm, the available computing power, and the sensor (e.g., lidar, camera, event camera, RGB-D camera), the perception latency can vary from tens up to hundreds of milli-seconds [22, 151, 120, 152, 10, 79].

At the current state of the art, the agility of autonomous robots is bounded, among the other factors (such as their actuation limitations), by their sensing pipeline. This is because the relatively high latency and low sampling frequency limit the aggressiveness of the control strategies that can be implemented. It is typical in current robots to have latencies of tens or hundreds of milli-seconds. Faster sensing pipelines can lead to more agile robots.

Despite the importance of the perception latency, very little attention has been devoted to study its impact on the agility of a robot for a sense and avoid task. Analyzing the role of sensing latency allows one to understand the limitations of current perception systems, as well as to comprehend the benefits of exploiting novel image sensors and processors, such parallel visual processors (e.g., SCAMP [67]), with a theoretical latency of few milli-seconds, or event cameras, with a theoretical latency of micro-seconds (e.g., the DVS [95]) or even nano-seconds (e.g., CeleX [68]).

In the context of robot navigation, it is also important to correlate the sensing latency to the actuation capabilities of the robot. Broadly speaking, the larger the acceleration a robot can produce, the lower the time it needs to avoid an obstacle and, therefore, the larger the latency it can tolerate. Consequently, the coupling between sensing latency and the actuation limitations of a robot represents a key research problem to be

addressed.

C.1.1 Related Work

Sensing latency is a known issue in robotics and has already been investigated before. For example, this problem is particularly interesting when the state estimation process is done through visual localization. A number of vision-based solutions for low-latency localization based either on standard cameras [56, 124] or novel sensors (e.g., event cameras [22, 155, 198]) have been proposed. Impressive results have been achieved, however no information about the environment is available since visual localization only provides the robot the information about its pose.

It is not yet clear what the maximum latency of a perception system for a navigation task should be. A first step in that direction is available in [69], where the authors studied under which circumstances a high frame-rate is best for real-time tracking, providing quantitative results that help selecting the optimal frame-rate depending on required performance. The results of that work were tailored towards visual localization for state estimation. In [187] the performance of visual servoing as a function of a number of parameters describing the perception system (e.g., frame-rate, latency) was studied, and a relation between the tracking error in the image plane and the latency of the perception was derived.

In [11], a framework to predict and compensate for the latency between sensing and actuation in a robotic platform aimed at visually tracking a fast-moving object was proposed and experimental results showed the benefits of that framework. Nevertheless, the impact of the latency on the performance of the executed task without the proposed compensation framework was not discussed.

The most similar work to ours is [166], where the authors studied the performance of vision-based navigation for mobile robots depending on the latency and the sensing range of the perception system. A trade-off among camera frame rate, resolution, and latency was shown to represent the best configuration for navigation in unstructured terrain. However, such results were only supported by experimental results, without any theoretical evidence. Different from our work, the actuation capabilities of the robot were not considered.

To the best of our knowledge, no previous works analyzed the coupling between sensing latency and actuation limitations in a robotic platform from a theoretical perspective. Similarly, the problem of highlighting their impact on the performance of high-speed navigation has not been addressed in the literature.

C.1.2 Contributions

In this work, we focus on the effects of perception latency and actuation limitations on the maximum speed a robot can reach to safely navigate through an unknown, static scenario.

We consider the case where a generic robot, modeled as a linear system with bounded inputs, moves in a plane and relies on onboard perception to detect static obstacles along its path (cf. Fig. C.1). We focus on a scenario where the robot wants to reach a target destination in as little time as possible, and therefore cannot change its longitudinal velocity to avoid obstacles. We show how the maximum latency the robot can tolerate to guarantee safety is related to the desired speed, the agility of the platform (e.g., the maximum acceleration it can produce), as well as other perception parameters (e.g., the sensing range). Additionally, we derive a closed-form expression for the maximum speed that the robot can reach as a function of its perception and actuation parameters, and study its sensitivity to such parameters.

We provide a general analysis that can serve as a baseline for future quantitative reasoning for design trade-offs in autonomous robot navigation, and is completely agnostic to the sensor and robot type. As a particular case study, we compare standard cameras against event cameras for autonomous quadrotor flight, in order to highlight the potential benefits of these novel sensors for perception. Finally, we provide an experimental evaluation and validation of the proposed theoretical analysis for the case of a quadrotor, equipped with an event camera, avoiding a ball thrown towards it at speeds up to $9\,\mathrm{m\,s^{-1}}$.

To the best of our knowledge, this is the first work in which perception and actuation limitations are jointly considered to study the performance of a robot in high-speed navigation.

C.1.3 Assumptions

This works is based on the following assumptions. First, we assume that the robot can be model as a linear system. Robotic systems are typically characterized by non-linear models. However, a large variety of them can be linearized through either static or dynamic feedback [176], rendering them equivalent from a control perspective to a chain of integrators. It is important to note that feedback linearization is different from Jacobian linearization: the first is an exact representation of the original non-linear system over a large variety of working conditions, while the second is only valid locally [71]. Linear models for mobile robots have already been used in the past [83], and come with the advantage of allowing a simple, yet effective mathematical analysis of the behaviour of the system in closed-form. Also, they cover a large variety of systems, rendering our analysis valid for different kinds of robots.

Second, we assume that the robot can execute holonomic 2D maneuvers. For non-holonomic systems, such as fixed-wing aircraft, the coupling of the longitudinal and lateral dynamics would break the assumptions of our model and would deserve a different analysis.

Finally, since we are interested in the role of sensing latency and actuation limitations on the agility of a robot, we assume that, for any other aspect, the sensing and actuation system are ideal. In other words, we assume that there is no uncertainty in the obstacle detection, no illumination issues, no artifacts in the measurements, and the robot's dynamics is perfectly known and can be controlled with errors. This allows us to clearly isolate and analyze the impact of sensing latency and actuation limitations in our analysis, where otherwise it would not be possible to distinguish the role of these two from the impact of other sources of non-ideality.

C.1.4 Structure of the Paper

In Sec. C.2, we provide the mathematical formulation of the problem and perform a qualitative analysis. In Sec. C.3, we particularize our study to vision-based navigation and analyze it for both standard and event cameras. A detailed mathematical analysis of these sensors is provided in the supplementary material. In Sec. C.4, we compare standard cameras (monocular and stereo) against event cameras for the case study of autonomous quadrotor flight. In Sec. C.5, we validate our analysis performing experiments on an actual quadrotor avoiding obstacle thrown towards it. Further details about the experiments are provided in the supplementary material. Finally, in Sec. C.6, we draw the conclusions.

C.2 Problem Formulation

We consider the case of a mobile robot navigating in a plane, which covers a large number of scenarios, e.g. an aerial robot flying in a forest [83], where the third dimension would not help with the avoidance task. The robot moves along a desired direction with a desired speed, provided by a high-level planner, towards its goal, which has to be reached in as little time as possible. Therefore, the robot cannot change its longitudinal velocity. In the following analysis, we consider the case where the robot only faces one single obstacle along its path and then provide an intuitive explanation of how our conclusions can be extended to the case of multiple obstacles.

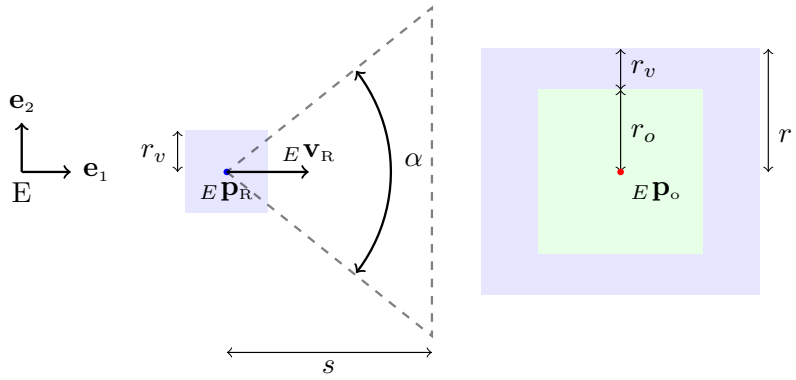


Figure C.1: A schematics representing the obstacle and the robot model in the frame E. The robot is represented as a square of size $2r_v$ centered at ${}_E\mathbf{p}_\mathrm{R}$, and moves with a speed ${}_E\mathbf{v}_\mathrm{R}$. The dashed triangle starting from the robot's position represents its sensing area, α is the field of view and s the maximum distance it is able to perceive. The obstacle, represented by the green square on the right side of the image, has size $2r_o$. We expand the square representing the obstacle by a quantity r_v such that the robot can be considered to be a point mass.

C.2.1 Modelling

Robot Model

Let E be the inertial reference frame, having basis $\{\mathbf{e}_1, \mathbf{e}_2\}$, and let ${}_E\mathbf{p}_\mathrm{R}$ and ${}_E\mathbf{v}_\mathrm{R}$ be the position and velocity, respectively, of the robot in E. Also, let ${}_E\mathbf{p}_\mathrm{O}$ be the position of an obstacle in E. In the remainder, we will refer to $\mathbf{e}_1$ as the *longitudinal* axis, and $\mathbf{e}_2$ as the *lateral* axis. Finally, let r_v be the half-size of the square centered at ${}_E\mathbf{p}_\mathrm{R}$ containing the entire robot (cf. Fig. C.1).

We model both the longitudinal and lateral dynamics as a chain of integrators. As shown in [176], a large variety of mechanical systems can be linearized by using nonlinear feedback, which, from a control perspective, renders them equivalent to a chain of integrators. Additionally, the dynamics of the actuators is usually faster than the mechanical dynamics and can, therefore, be neglected.

The longitudinal and lateral dynamics are modeled by a position $\mathrm{p_i}$, a speed $\mathrm{v_i}$ and an input $\mathrm{u_i}$ given by:

$$\dot{\mathrm{p}}_1(t) = \mathrm{v}_1(t), \quad \dot{\mathrm{v}}_1(t) = \mathrm{u}_1(t), \tag{C.1}$$

$$\dot{\mathrm{p}}_2(t) = \mathrm{v}_2(t), \quad \dot{\mathrm{v}}_2(t) = \mathrm{u}_2(t). \tag{C.2}$$

Both inputs are assumed to be bounded such that $\mathrm{u_i} \in [-\bar{\mathrm{u}}_\mathrm{i}, \bar{\mathrm{u}}_\mathrm{i}], i = 1, 2$. We assume

the robot to move only along the longitudinal axis with an initial speed $v_{1,0} = \hat{v}_1$, meaning that the lateral speed is zero before the avoidance maneuver starts. The case where the robot has non-zero lateral velocity can be analyzed using the same mathematical framework. Also, we assume that the robot cannot change its longitudinal speed, namely $u_1(t) = 0 \; \forall t$, and can therefore only exploit the lateral dynamics to avoid an obstacle. As shown in Sec. C.2.1 of the supplementary material, a lateral avoidance maneuver requires less time at high speed, allowing faster navigation along the longitudinal axis.

Obstacle Avoidance: Brake or Avoid?

To avoid an obstacle, a robot can either stop before colliding or circumvent it by moving laterally. Fig. C.2 shows a comparison between (i) the minimum time $t = \frac{\hat{v}_1}{\bar{u}_1}$ required for a robot to brake and stop before colliding, and (ii) the minimum time required to avoid the obstacle laterally without braking (see Sec. C.2.2). We considered $\bar{u}_1 = \bar{u}_2 = 25\,\mathrm{m/s^2}$, and the horizontal axis reports the longitudinal speed towards the obstacle. The results show that the lateral avoidance maneuver requires less time at high speed, allowing faster navigation along the longitudinal axis. Additionally, a continuous motion along the desired direction is preferable over a *stop-avoid-go* behaviour, since would allow the robot to navigate faster and reach its goal earlier. Therefore, we consider only the case where the robot does not brake to prevent the collision, but rather executes a lateral avoidance maneuver.

Obstacle Model

We consider static obstacles enveloped by a square of width $2r_o$. To study the motion of the robot considering only the position of its center, we expand the obstacle width by a quantity r_v on each side. The expanded size of the obstacle is $r = 2(r_v + r_o)$, as shown in Fig. C.1.

Sensor Model

In this work, we assume that at least one edge of the obstacle must enter the sensing area to allow a detection. We define the sensing latency $\tau \in \mathbb{R}^+$ as the interval between the time the obstacle enters the sensing area and the moment the robot's initiates the avoidance maneuver. The latency of a sensor is typically the sum of multiple contributions, and in general depends on the sensor itself and the time necessary to process a measurement (which depends on the algorithm used, the computational power available, and other factors). In general, it is hard to provide exact bounds for each of these contributions, therefore we consider as latency the sum of the sensor's and the sensing algorithm's latency. We denote by $s \in \mathbb{R}^+$ the robot's sensing range,

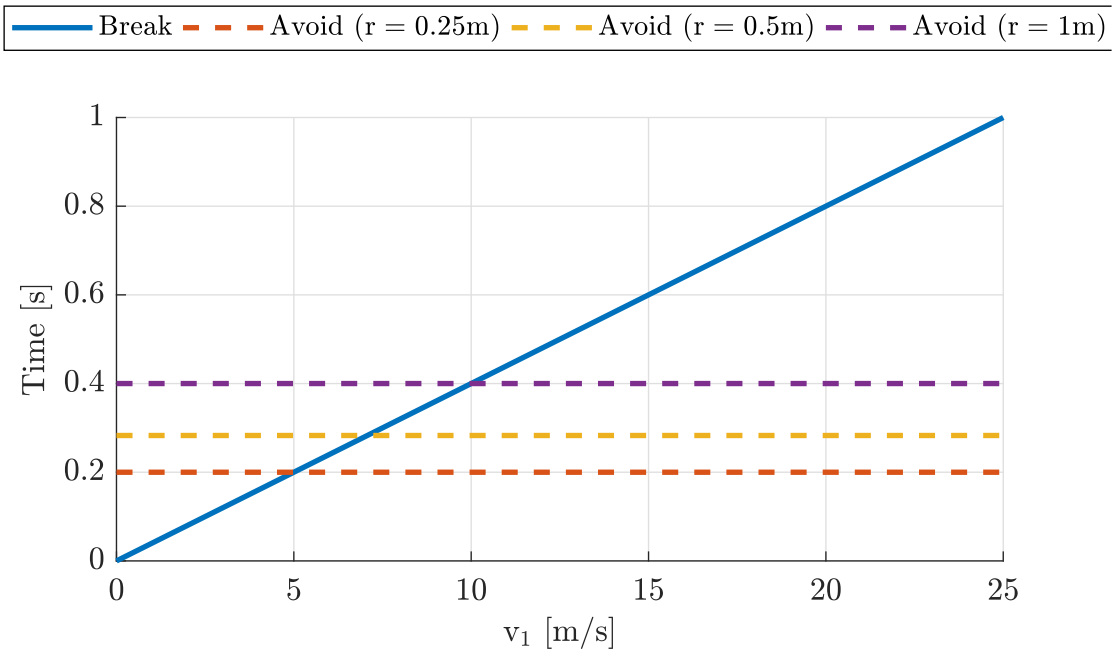


Figure C.2: Comparison between the minimum time required for a robot to completely stop before colliding (solid blue line) and the minimum time required to move laterally by an amount r (dashed lines), depending on the speed $\hat{v}_1$ (horizontal axis).

i.e. the largest distance it is able to perceive. We assume the field of view of the sensor to be such that the obstacle's edge is fully contained in the sensing area when the distance between the robot and the obstacle is equal to the sensing range. This provides a lowerbound for the field of view $\alpha \geq 2 \arctan\left(\frac{r_o}{2s}\right)$.

C.2.2 Obstacle Avoidance

Time to Contact and Avoidance Time

We define the *time to contact* t_c as the time it takes the vehicle to collide with the obstacle once it enters the sensing range of its onboard sensor. Since the longitudinal motion has a constant speed $\hat{v}_1$ and the distance between the vehicle and the obstacle at the time the obstacle enters the sensing area is s, the time to contact t_c is:

$$t_c = \frac{s}{\hat{v}_1}. \tag{C.3}$$

In order for the robot to avoid the obstacle, it has to reach a safe lateral position in an *avoidance time* t_s shorter than the time to contact (C.3).

$$t_c \geq t_s. \tag{C.4}$$

Time-Optimal Avoidance

The avoidance maneuver along the lateral axis leads to a safe navigation if $p_2(t_c) \geq r$. We consider the case $p_2(t_c) = r$, which represents the minimum lateral deviation for the avoidance maneuver to be executed safely. For this to happen, we assume the robot to use a time-optimal strategy $u_2^*(t)$:

$$\begin{aligned} u_2^*(t) = \arg\min_{u_2(t)} \quad & t_s \\ \text{subject to} \quad & \dot{p}_2(t) = v_2(t),\ \dot{v}_2(t) = u_2(t), \\ & p_2(0) = 0, \quad v_2(0) = 0, \\ & p_2(t_s) = r, \quad v_2(t_s) = 0, \\ & u_2(t) \in [-\bar{u}_2, \bar{u}_2]\ \forall t. \end{aligned} \tag{C.5}$$

We require $v_2(t_s) = 0$ because there would be no advantage in having a non-zero lateral speed in terms of progressing towards the goal, since we considered the longitudinal axis to be the direction of motion. Leaving the final lateral speed free would lead to a lower execution time for the avoidance maneuver, but this could potentially result in a large lateral speed, which is typically not desirable because the robot is not able to sense the environment in such a direction. As well known in the literature [12], the problem (C.5) leads to a *bang-bang* solution:

$$u_2^*(t) = \begin{cases} \bar{u}_2 & \text{if } 0 \leq t \leq \hat{t} \\ -\bar{u}_2 & \text{if } \hat{t} < t \leq t_s \end{cases}, \tag{C.6}$$

where the $\hat{t} = \sqrt{\frac{r}{\bar{u}_2}}$ is the switching time and $t_s = 2\sqrt{\frac{r}{\bar{u}_2}}$ is the avoidance time..

Obstacle Avoidance with Sensing Latency

In Sec. C.2.2 we defined the time to contact t_c as the time between when the obstacle enters the sensing range and the moment when the collision occurs, as defined in (C.3). However, in the presence of sensing latency, the time $t_c^{'}$ remaining to the collision when the robot is informed about the presence of the obstacle is $t_c^{'}(\tau) = t_c - \tau$. Therefore, in order for a robot equipped with a sensor with sensing range s and latency τ to safely avoid an obstacle, the condition $t_c^{'}(\tau) \geq t_s$ must hold. In this case, we can compute (C.4) as:

$$\frac{s}{\hat{v}_1} - \tau \geq 2\sqrt{\frac{r}{\bar{u}_2}}. \tag{C.7}$$

The worst case in which the robot manages to avoid the obstacle occurs when (C.7) is

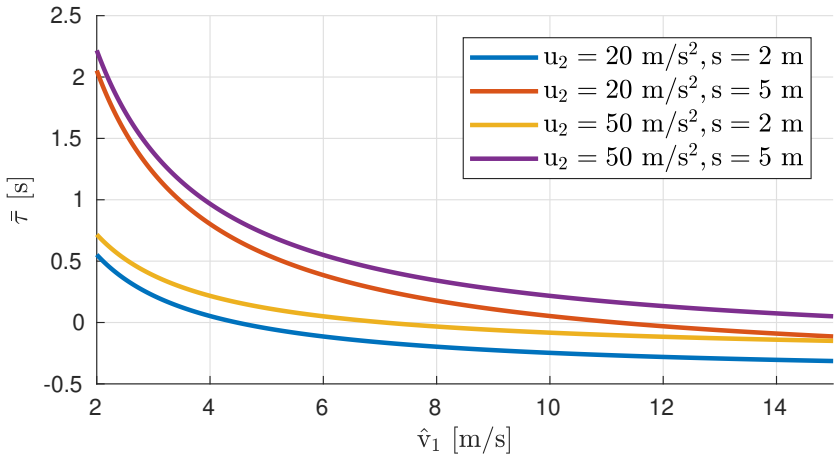


Figure C.3: Maximum latency $\bar{\tau}$ that the robot can tolerate in order to safely perform the avoidance maneuver when $r = 0.5\,\text{m}$.

satisfied with equality. In this case, the robot passes tangent to the obstacle, whereas it would have some safety margin if (C.7) was satisfied with the inequality sign. We can study (C.7) to compute the maximum latency $\bar{\tau}$ the system can tolerate such that the avoidance can still be performed safely:

$$\bar{\tau} = \frac{s}{\hat{v}_1} - 2\sqrt{\frac{r}{\bar{u}_2}}. \tag{C.8}$$

Fig. C.3 shows the maximum latency $\bar{\tau}$ for different values of $\bar{u}_2$ and s for the case $r = 0.5\,\text{m}$. As one can notice, the importance of low latency increases as the navigation speed increases. Also, for some speeds $\hat{v}_1$ the robot is unable to perform the avoidance maneuver safely given its actuation capabilities and the sensing range of its sensor. This is clear from the negative values the maximum latency $\bar{\tau}$ assumes in some intervals. In this case the robot should be either more agile (i.e. capable of generating higher lateral accelerations) or should be equipped with a sensor with a higher sensing range in order to avoid the obstacle at such speeds.

Similarly, we can use (C.8) to compute the maximum longitudinal speed the robot can have to avoid the obstacle:

$$\bar{v}_1 = \frac{s}{\tau + 2\sqrt{\frac{r}{\bar{u}_2}}}. \tag{C.9}$$

Fig. C.4 shows the maximum speed the robot can navigate safely (i.e., being still able to avoid the obstacle although this is perceived with some delay), depending on the latency of its sensing pipeline.

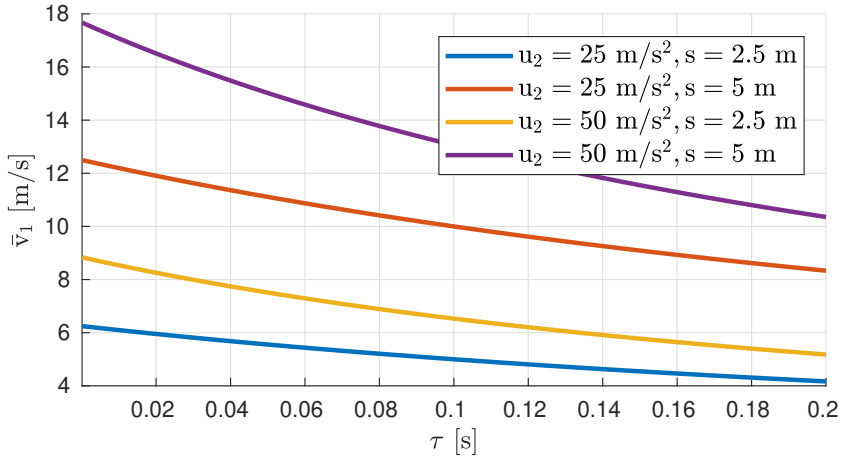


Figure C.4: Maximum speed $\bar{v}_1$ that the robot can move in order to safely perform the avoidance maneuver when $r = 0.5$ m.

C.3 Vision-Based Perception

In the following, we particularize our analysis to the case of vision-based perception for three modalities: (i) a monocular frame-based camera, (ii) a stereo frame-based camera, (iii) a monocular event camera, and analyze the impact of their latency on the maximum speed. For brevity reasons, the mathematical derivation of the expressions for the sensing range and the latency of each of these sensing modalities is reported in the supplementary material attached to this work.

C.3.1 Frame-Based Cameras and Event Cameras

Most computer vision research has been devoted to frame-based cameras, which have latencies in the order of tens of milli-seconds, thus, putting a hard bound on the achievable agility of a robotic platform. By contrast, event cameras [95] are bio-inspired vision sensors that output pixel-level brightness changes at the time they occur, with a theoretical latency of micro-seconds or even nano-seconds. More specifically, rather than streaming frames at constant time intervals, each pixel *fires* an event (a pixel-level brightness change), independently of the other pixels, every time it detects a change of brightness in the scene. Broadly speaking, we can consider event cameras as *motion-activated*, asynchronous edge detectors: events fire only if there is relative motion between the camera and the scene.

Exploiting frame-based cameras for obstacle avoidance typically requires the analysis of all the pixels of the image to detect an obstacle, independently of the texture. Conversely, since the pixels of an event camera only trigger information when there is change of intensity, it has the advantage of requiring very little processing to detect an

obstacle. Furthermore, since the smallest time interval between two consecutive events on the same pixel is in the order of 1 µs, or generally much smaller than the typical framerate of frame-based cameras, this can safely be neglected. These factors result in a theoretical advantage of event cameras against frame-based cameras.

C.3.2 Sensing Range of a Vision-Based Perception System

Monocular Frame-Based Camera

The sensing range s_{M} of a monocular camera depends, as shown in Sec. C.9.1 of the supplementary material, on the size r_o of the obstacle, the number of pixels N it must occupy in the image to be detected, and the camera's angular resolution θ.

Stereo Frame-Based Camera

The sensing range s_{S} of a stereo camera depends, as shown in Sec. C.10.1 of the supplementary material, on the baseline b, the focal length f, the uncertainty in the disparity ϵ_{P} and the maximum percentual uncertainty k in the depth estimation.

Event Camera

In Sec. C.11.1 of the supplementary material we show that the sensing range s_{E} of an event camera can be computed using (C.10). It depends on how large the object must be in the image such that, when its edges generate an event, they are sufficiently far apart.

C.3.3 Latency of a Vision-Based Perception System

Monocular Frame-Based Camera

The latency τ_{M} of a monocular camera depends on the time t_{f} between two consecutive triggers of the sensor, the exposure time t_{E}, the transfer time t_{T}, the processing time and the number of images necessary to detect the obstacle. As shown in Sec C.9.2 of the supplementary material, if two consecutive images are sufficient to detect an obstacle, it can vary between $\tau_{\mathrm{M}} = t_{\mathrm{f}} + t_{\mathrm{T}} + t_{\mathrm{E}}$ and $\tau_{\mathrm{M}} = 2t_{\mathrm{f}}$.

Stereo Frame-Based Camera

In Sec. C.10.2 of the supplementary material, we analyze the possible range of the latency τ_{S} of a stereo camera. In general, it can span between a best-case value equal to the time between two consecutive frames, and a worst-case value, which we derive

analyzing the datasheet of several stereo cameras.

Event Camera

The latency τ_E of an event camera depends, as shown in Sec. C.11.2 of the supplementary material, on the distance between the camera and the obstacle, the speed of the camera, the focal length, and the amount of pixels the projection of the obstacle must move in the image such that it fires an event. However, to derive the maximum speed achievable with an event camera, it is necessary to jointly consider the expression of the latency of an event camera and (C.4). We refer the reader to Sec. C.11.2 of the supplementary material for further details.

C.4 Case Study: Vision-Based Quadrotor Flight

In this section, we analyze the case of vision-based quadrotor flight. We consider a quadrotor equipped with a sensing pipeline based on frame-based cameras in a monocular and stereo configuration, and a monocular event camera. For each sensing modality, we provide an upper and a lower-bound of the sensing range and the latency according to the model in Sec. C.3. We compute the maximum speed achievable with each sensor for a value of each parameter equal to its lower-bound, its upper-bound, and the average between the upper and the lower-bound. Finally, we consider four different values for the maximum lateral acceleration the quadrotor can produce. Three values correspond to commercially available state-of-the-art quadrotors with low, medium and high *thrust-to-weight* ratio. The fourth one, instead, represents a quadrotor with a *thrust-to-weight* ratio that is, as of today, particularly hard to achieve with current technology, but might become common in the future. This ideal platform serves us to show that more agile quadrotors would significantly highlight the benefits of lower-latency sensors for obstacle avoidance.

C.4.1 Sensing Range

Monocular Frame-Based Camera

We use the results of Sec. C.9.1 of the supplementary material to obtain the upper-bound and the lower-bound for the sensing range of a monocular camera. The best-case scenario occurs when the obstacle to be detected occupies 5% of the image, leading to an upper-bound $s_M = 6\,\text{m}$. We consider as worst-case scenario when the obstacle occupies 10%, leading to a lower-bound $s_M = 2\,\text{m}$.

Stereo Frame-Based Camera

We assume the robot to be equipped with a stereo system having a baseline $b = 0.10\,\text{m}$ and each camera having a VGA resolution. As shown in Sec. C.10.1 of the supplementary material, we consider $s_S = 2\,\text{m}$ and $s_S = 8\,\text{m}$ to be reasonable values for the lower-bound and the upper-bound of the sensing range.

Event Camera

As mentioned in Sec. C.11.1 of the supplementary material, the sensing range of an event camera can reach values above $s_E = 10\,\text{m}$. Intuitively speaking, this is because to potentially detect an obstacle with an event camera, it is sufficient that the projection of its edges move on the image by 1 pxl and are far apart from each others by an amount that is at least on order of magnitude larger (i.e., at least 10 pxl apart). However, to render our comparison more fair and realistic, we consider a lower-bound that is comparable to the one of frame cameras. Indeed, when a robot navigates cluttered environments, its distance from the obstacles is typically lower than 10 m, which makes it necessary to consider a lower value for the smallest sensing range of event camera. Therefore, we assume $s_E = 2\,\text{m}$ as lower-bound for the sensing range of an event camera, and $s_E = 8\,\text{m}$ as its upper-bound.

C.4.2 Latency

Monocular Frame-Based Camera

We consider a frame-based camera with (i) a framerate of 50 Hz, meaning that $t_f = 0.020\,\text{s}$; (ii) an exposure time of $t_E = 0.005\,\text{s}$; (iii) VGA resolution and USB 3.0 connection, which leads to $t_T = 0.000{,}4\,\text{s}$. Therefore, based on Sec. C.9.2 of the supplementary material, the upper-bound and the lower-bound latency for the frame-based camera considered in this analysis are, respectively, $\tau_M = 0.040\,\text{s}$ and $\tau_M = 0.026\,\text{s}$.

Stereo Frame-Based Camera

As mentioned in Sec. C.10.2 of the supplementary material, it is hard to evaluate the latency of a stereo system. However, based on the datasheet of commercially available stereo cameras suitable for quadrotor flight, we can obtain an estimate of the upper-bound and the lower-bound. As upper-bound, we consider the Bumblebee XB3, whose datasheet reports a latency of $\tau_S = 0.070\,\text{s}$. For the lower-bound, since no further information are available in the datasheet of other stereo cameras, we assume it to be equal to the inverse of the frame-rate of the fastest available sensor (Intel RealSense R200) leading to $\tau_S = 0.017\,\text{s}$.

Event Camera

In Sec. C.11.2 of the supplementary material we discuss how the latency of an event camera depends on the relative distance and speed between the robot and the obstacle. Also, we highlight that, in order to compute it, it is necessary to jointly consider the sensing range (Sec. C.11.1), Eq. (C.8) and Eq. (C.14). Therefore, to analyze the maximum speed achievable with an event camera we proceed as follows: (i) we consider a value of the sensing range as described in Sec. C.3.2; (ii) we plug (C.9) into (C.14) and solve it for $\hat{v}_1$ to compute the maximum speed achievable; (iii) we use (C.8) to obtain the corresponding value of the latency of an event camera, given its distance from the obstacle and its speed.

C.4.3 Quadrotor Model

The dynamical model of a quadrotor is differentially flat and the vehicle can be considered as a linear system using nonlinear feedback linearization [110] both from a control [101] and a planning perspective [130]. We considered four cases for the maximum lateral acceleration the robot can produce: $\bar{u}_2 = 10\,\mathrm{m\,s^{-2}}$, $\bar{u}_2 = 25\,\mathrm{m\,s^{-2}}$, $\bar{u}_2 = 50\,\mathrm{m\,s^{-2}}$, and $\bar{u}_2 = 200\,\mathrm{m\,s^{-2}}$. These values correspond to a *thrust-to-weight ratio* of approximately 1.5, 2.8 5.2 and 20, respectively. The first three cover a large range of the lift capabilities of commercially available drones, while the fourth represents a vehicle currently not yet available, but which might be available in the future. We assume $r_v = 0.25\,\mathrm{m}$ and $r_o = 0.50\,\mathrm{m}$, leading to an expanded obstacle size of $r = 0.75\,\mathrm{m}$.

C.4.4 Results

The results of our analysis for vision-based quadrotor flight are available in Table C.1. For each sensing modality (first column) we combined three values for the sensing range (second column) and the latency (third column), and computed the maximum speed the robot can achieve depending on the maximum lateral acceleration it can produce (fourth column). For frame-based camera (monocular and stereo), we considered as values for the sensing range and the latency the lower-bound, the upper-bound and the average between upper-bound and lower-bound.

Similarly, we considered three values for the sensing range of an event camera. However, as mentioned in Sec. C.4.2, the latency of event cameras is strictly connected to the robot's agility. As shown in Sec. C.11.2 of the supplementary material, the theoretical latency of an event camera depends on both its distance to the obstacle and its velocity towards it (c.f. Eq. (C.14)). Broadly speaking, the faster the robot, the earlier the desired amount of events for the detection are generated. However, for the obstacle avoidance problem to be well-posed, the robot cannot be arbitrarily fast, but its speed must be such that the avoidance maneuver requires an amount of time smaller than

Sensor Type	Sensing Range [m]	Latency [s]	Max. speed [m s^{-1}] $\bar{u}_2 = 10\,\mathrm{m\,s^{-2}}$	$\bar{u}_2 = 25\,\mathrm{m\,s^{-2}}$	$\bar{u}_2 = 50\,\mathrm{m\,s^{-2}}$	$\bar{u}_2 = 200\,\mathrm{m\,s^{-2}}$
Mono Frame	2.0	0.026	3.48	5.37	7.38	13.47
	2.0	0.033	3.44	5.27	7.20	12.83
	2.0	0.040	3.40	5.17	7.02	12.30
	4.0	0.026	5.23	8.06	11.07	26.94
	4.0	0.033	5.17	7.91	10.79	25.73
	4.0	0.040	5.10	7.76	10.53	24.62
	6.0	0.026	6.97	10.74	14.76	40.41
	6.0	0.033	6.89	10.54	14.39	38.59
	6.0	0.040	6.81	10.35	14.03	36.93
Stereo Frame	2.0	0.017	3.54	5.51	7.64	14.37
	2.0	0.043	3.38	5.13	6.93	12.06
	2.0	0.070	3.24	4.80	6.35	10.39
	5.0	0.017	8.86	13.77	19.11	35.93
	5.0	0.043	8.50	12.83	17.34	30.16
	5.0	0.070	8.10	12.01	15.88	25.98
	8.0	0.017	14.17	22.03	30.57	57.50
	8.0	0.043	13.54	20.53	27.75	48.25
	8.0	0.070	12.95	19.21	25.40	41.56
Mono Event	2.0	0.002	-	-	-	16.12
	2.0	0.003	-	-	8.06	-
	2.0	0.004	-	5.70	-	-
	2.0	0.007	3.60	-	-	-
	5.0	0.004	-	-	-	39.53
	5.0	0.008	-	-	19.76	-
	5.0	0.011	-	13.98	-	-
	5.0	0.017	8.84	-	-	-
	8.0	0.006	-	-	-	62.06
	8.0	0.012	-	-	31.03	-
	8.0	0.018	-	21.94	-	-
	8.0	0.029	13.88	-	-	-

Table C.1: The results of our case study. We compare monocular frame-based cameras, stereo frame-based cameras and event cameras for different robot agility values. The dashes in the columns reporting the maximum speed achievable with an event camera are due to the fact that, given a value for the sensing range and the maximum lateral acceleration, we can compute the maximum achievable speed and the corresponding latency (c.f. Sec. C.4.4 for a more detailed explanation).

the time to contact (Eq. (C.4) and (C.7)). This means that the theoretical latency of an event camera depends also on the maximum lateral input the robot can produce. Therefore, for a given sensing range and robot's maximum input, one can compute the corresponding maximum velocity achievable and, consequently, the latency of an event camera mounted on such a robot. Since different robot's maximum input would produce different maximum velocity, the same event camera will similarly have different latencies (Eq. (C.14)). This motivates the dashed values in Table C.1.

As one can notice, when the sensing range and the robot's agility are small, the difference among mononocular frame cameras, stereo frame cameras and event cameras is not remarkable. Conversely, frame cameras in stereo configuration and event cameras allow faster flight than a monocular frame camera when either the sensing range or the robot's agility increase. In particular, increasing the sensing range, as expected from Sec. C.7, allows the robot to navigate faster thanks to a sensible increase of the time to contact.

Similarly, making the robot more agile (i.e., increasing $\bar{u}_2$) allows it to fly faster thanks to the decrease of the avoidance time. As one can notice by the results in the column of the quadrotor having and $\bar{u}_2 = 200\,\mathrm{m\,s^{-2}}$, the difference between the maximum speed achievable with stereo frame-based cameras and event cameras become significant. Depending on the sensing range, low-latency event cameras allow the robot to reach a maximum speed that can be between 7% and 12% larger than the one achievable with a stereo frame-based camera. It is important to remark that, despite the numbers provided for the case $\bar{u}_2 = 200\,\mathrm{m\,s^{-2}}$ are very high, they are not as far as one could think from what is currently achievable by agile quadrotors. Indeed, First-Person-View (FPV) quadrotors are currently capable of reaching speeds above $40\,\mathrm{m\,s^{-1}}$ with thrust-to-weight ratios above 10 and, given the pace of the technological progress in the FPV community, it is not hard to believe that, in the near future, quadrotors will be able to reach speeds significantly beyond the current values. In FPV racing, a small increase in the maximum flight speed can represent the step necessary to outperform other vehicles participating in the race. This is particularly interesting in the contest of autonomous FVP drone racing, an extremely active area of research [84, 165].

C.5 Experiments

To validate our analysis, we performed real-world experiments with a quadrotor platform equipped with an Insightness SEEM1 sensor [1], a very compact neuromorphic camera providing standard frame, events and Inertial Measurement Unit data. The obstacle was a ball of radius 10 cm thrown towards the quadrotor, and the vehicle only relied on the onboard event camera to detect it and avoid it. From the perspective of our model, this is equivalent to the case where the robot moves towards the obstacle, since the time to contact depends on the absolute value of the relative longitudinal velocity. This experimental setup allowed us to reach large relative velocities in a confined space. Further details about the experimental platform used in this work are available in Sec.C.13.1 of the supplementary material.

C.5.1 Obstacle Detection with an Event Camera

To detect the obstacle, whose size is supposed to be known, we use a variation of the algorithm proposed in [119] to remove events generated by the static part of the environment due to the motion of the camera. Different from [119], we do not compensate for the camera's motion using numerical optimization, but rather exploiting the gyroscope's measurements. This allows our pipeline to be faster, but comes at the cost of a higher amount of not compensated events.

We accumulate motion-compensated events over a sliding window of 10 ms, obtaining

[1] http://www.insightness.com/technology

an *event-frame* containing the timestamp of the events due to the motion of moving objects. Such event-frame typically consists of several separated blobs, which are clustered together using the DBSCAN algorithm [39] based on their relative distance, their direction of motion (obtained using Lucas-Kanade tracking [102]) and the timestamp of the events. We fit a rectangle around the blobs belonging to the same cluster and look for the rectangle having the most similar aspect ratio to the expected one. Since we assume the size of the obstacle to be known, we compute its expected aspect ratio and, after finding the most similar cluster, we project its the centroid into the world frame using the standard pinhole camera projection model.

To render our algorithm most robust to outliers, we considered the obstacle to be detected only when at least n measurements in the world frame are obtained and their relative distance is below a threshold. Our experimental evaluation showed that 2 consecutive measurements at a relative distance lower than 20 cm were sufficient to detect the ball in a reliable way. Also, we fixed the sensing range by discarding detections happening when the ball was at a distance from the robot larger than its sensing range.

It is important to note that our detection algorithm was designed with the aim of reducing the latency of the sensing pipeline and, during the tuning stage, speed was prioritized against accuracy. Accurate obstacle detection with event cameras of obstacles of unknown size and shape is beyond of the scope of this paper.

C.5.2 Expected and Measured Latency

Theoretically, a 1 pxl motion of the projection of point in the image is sufficient to generate an event. However, in our experiment we realized that a larger motion is necessary to obtain reliable obstacle detection with an event camera. More specifically, the algorithm was able to detect the obstacle thrown towards the vehicle whenever a displacement between of at least 5 pxl was verified. In Sec. C.13.3 of the supplementary material we analyze this aspect and discuss the main reasons causing the discrepancy between the theoretical ideal model and real data. Also, we exploited the model proposed in Sec. C.11.2 of the supplementary material to compute the theoretical latency for an event camera having the same resolution of the sensor used in our experiments, for a pixel displacement of 5 pxl. Sec. C.13.2 of the supplementary material reports the theoretical latency for an obstacle detection pipeline based on an Insightness SEEM1, and the measured latency for our algorithm. As one can see from Fig. C.12, Fig. C.13, and Tab. C.2 in the supplementary material, the experimental data agree with the theoretical model. Sec. C.13.3 of the supplementary material discusses the discrepancy between our model and actual data.

C.5.3 Results

We performed experiments where the quadrotor described in Sec. C.13.1 of the supplementary material, equipped with an Insightness SEEM1 sensor and running the detection algorithm described in Sec. C.5.1, was commanded to avoid a ball thrown towards it. The ball was thrown with a speed spanning between $\hat{\mathrm{v}}_1 = 5\,\mathrm{m\,s^{-1}}$ and $\hat{\mathrm{v}}_1 = 9\,\mathrm{m\,s^{-1}}$. The sensing range was 2 m, meaning that any detection at distance larger than this amount was neglected. Therefore, the time to contact spanned between $t_\mathrm{c} = 0.22\,\mathrm{s}$ and $t_\mathrm{c} = 0.40\,\mathrm{s}$. The robot was commanded to execute an avoidance maneuver either upwards, laterally or diagonally. The obstacle radius was $r_o = 10\,\mathrm{cm}$, while the robot's size was computed as either its height ($r_v = 15\,\mathrm{cm}$) or half its tip-to-tip diagonal ($r_v = 25\,\mathrm{cm}$), depending on the direction of the the avoidance maneuver. Therefore, the expanded obstacle radius spanned between $r = 25\,\mathrm{cm}$ and $r = 35\,\mathrm{cm}$. The avoidance spanned between $t_\mathrm{s} = 0.17\,\mathrm{s}$ and $t_\mathrm{s} = 0.25\,\mathrm{s}$. In all the experiments, the ball would have hit the vehicle if the avoidance maneuver was not executed, as confirmed by ground truth data provided by the motion-capture system.

C.6 Conclusions

In this work, we studied the effects that perception latency has on the maximum speed a robot can reach to safely navigate through an unknown environment. We provided a general analysis for a robot modeled as a linear second-order system with bounded inputs. We showed how the maximum latency the robot can tolerate to guarantee safety is related to the desired speed, the agility of the platform (e.g., the maximum acceleration it can produce), as well as other perception parameters (e.g., the sensing range). We compared frame-based cameras (monocular and stereo) against event cameras for quadrotor flight. Our analysis showed that the advantage of using an event camera is higher when the robot is particularly agile. We validated our study with experimental results on a quadrotor avoiding a ball thrown towards it a speeds up to $9\,\mathrm{m\,s^{-1}}$ using an event camera. Future work will investigate the use of event cameras for obstacle avoidance on a completely vision-based quadrotor platform, using on-board Visual-Inertial Odometry for state estimation.

Supplementary Material of:

How Fast is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid

Davide Falanga, Suseong Kim and Davide Scaramuzza

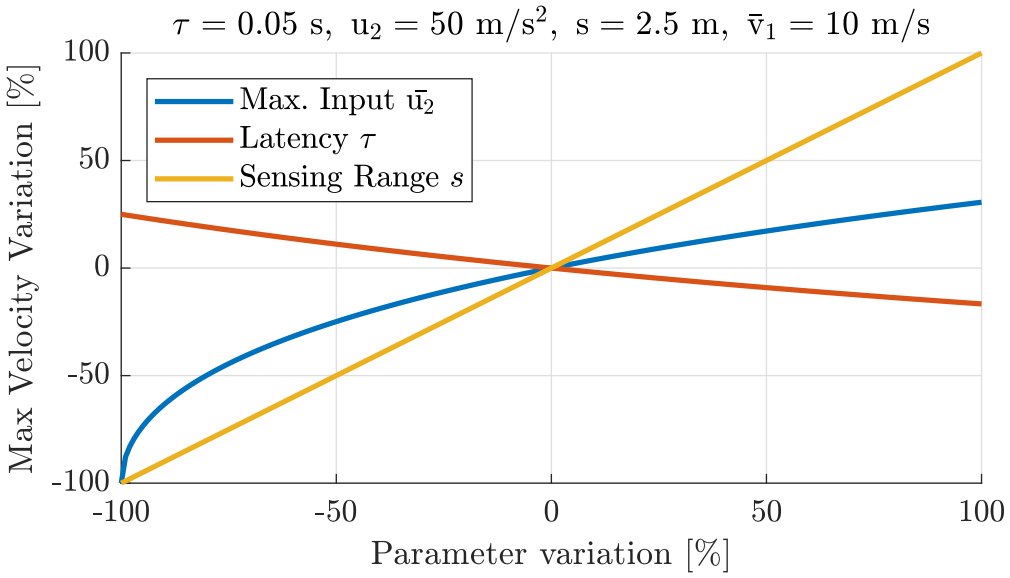


Figure C.5: Sensitivity of the maximum speed $\bar{v}_1$ with respect to the perception (s, τ) and actuation ($\bar{u}_2$) parameters of the system for the case $s = 2.5\,\text{m}$, $\tau = 0.05\,\text{s}$, $\bar{u}_2 = 50\,\text{m}\,\text{s}^{-2}$.

C.7 Sensitivity Analysis

Eq. (C.9) is particularly interesting for robot design to analyze what the best configuration in terms of perception and actuation systems is. As one can easily derive from (C.9), reducing the latency increases the maximum speed at which the robot can navigate the environment safely. However, it might not always be possible to reduce the sensing latency, or it might be better to change some other parameters of the system (e.g., the sensing range or the maximum acceleration), since this might produce better improvements at a lower cost. By performing a sensitivity analysis, we can study the impact of the sensing range, the latency, and the maximum input on the speed that the robot can reach.

To do so, it is necessary to first define a set of parameters. For example, we consider the case $s = 2.5\,\text{m}$, $\tau = 0.05\,\text{s}$, $\bar{u}_2 = 50\,\text{m}\,\text{s}^{-2}$. This set of parameters, chosen as a representative case for the study in Sec. C.4, according to (C.9) allow the robot to navigate at a maximum speed $\bar{v}_1 = 10\,\text{m}\,\text{s}^{-1}$. Based on these values, we vary each of the parameters while keeping the others constant to understand how the maximum speed the robot can achieve changes.

Fig. C.5 shows the results of this numerical analysis for a variation of the parameters between -100% and 100% of the reference value (horizontal axis). On the vertical axis the percentage variation of $\bar{v}_1$ is reported. As one can see, $\bar{v}_1$ is very sensitive to the sensing range, whereas, except for extreme decreases of the maximum lateral acceleration (far left end of the blue line), the sensitivities with respect to $\bar{u}_2$ and τ

are comparable. However, it is not always possible to change the range of a sensing pipeline, whereas it could be possible to reduce its latency. This is the case, for example, of a DAVIS [16], a neuromorphic sensor comprising a frame and an event camera sharing the same pixel array and optics. In such a case, it is possible to use frames or events depending on the need, but the sensing range, which depends on the sensor itself, cannot be modified for one modality without affecting the other. For this reason, in the remainder of this work will focus on the impact of the latency on the maximum speed a robot can navigate.

C.8 Generalization to Multiple Obstacles

So far, we only considered the case where the robot faces a single obstacle and needs to avoid it. Although mathematically simple, our approach can generalize to multiple obstacles by iteratively running the same considerations previously described. Independently of the number of obstacles, we can always consider the closest obstacle to the robot along its direction of motion and perform the evaluation of Sec. C.2.2 and C.2.2. If the robot reaches a safe lateral position within the time to contact (C.3), we can consider the obstacle *avoided*, and the robot has to avoid the next obstacle along its path. The only difference with respect to the previously avoided obstacle is the distance between the obstacle and the robot along the longitudinal and lateral axes.

A conservative, yet effective analysis can be conducted for the case of navigation in environments with multiple obstacles by using our formulation under the following assumptions: (i) all the obstacles are considered to have the same size (i.e., the size of the largest obstacle); (ii) the distance between two consecutive obstacles along the longitudinal axis is sufficiently large to guarantee that the avoidance time in the case of no latency is lower than the time to contact.

C.9 Monocular Frame-Based Camera

C.9.1 Sensing Range

For an obstacle to be detected with a frame-based camera, it has to occupy a sufficiently large number of pixels in the image. Let N be the number of pixels necessary to detect an obstacle. Furthermore, let α be the field of view of the sensor. Without loss of generality, we only consider the projection of an object along the horizontal axis of the camera, but similar results apply to the vertical axis.

Let q be the horizontal resolution of a camera. The angular resolution of the camera can be computed as $\theta = \frac{\alpha}{q}$. Let r_o be the size of an obstacle, d its distance to the camera, and assume it is placed such that the camera optical axis passes through its center

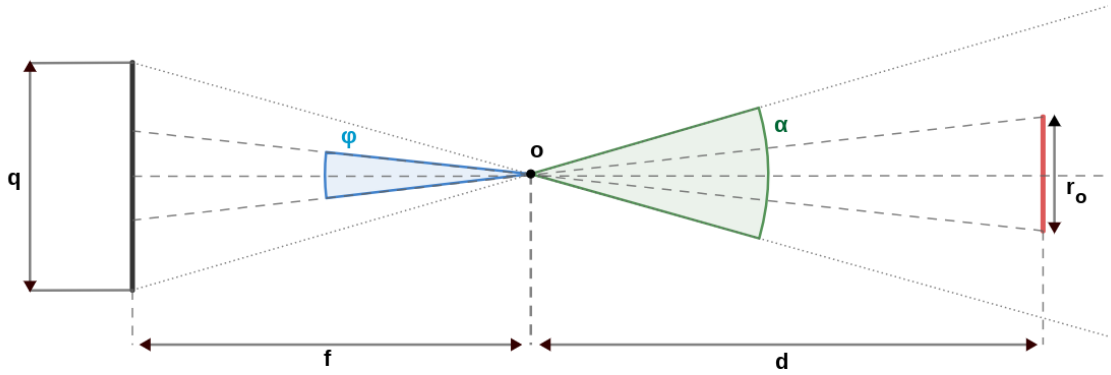


Figure C.6: A schematics representing the obstacle in front of the camera. The obstacle is represented in red, while the camera is in black on the left side of the image and has resolution q. The field of view α is highlighted in green, while the angle spanned by the obstacle in the image ϕ is highlighted in blue. d is the distance between the camera and the obstacle, while f is the focal length of the camera.

(cf. Fig. C.6). The obstacle spans an angle $\phi = 2\arctan\left(\frac{r_o}{2d}\right)$. For the obstacle to be visible in the image, it must be at a distance d such that $\phi = \theta$, which would result in a projection in the image of 1 pxl. However, 1 pxl is typically not sufficient to detect an obstacle. Let N be the number of pixels one needs to detect an obstacle. For the obstacle to occupy at least N pixels in the image, we want that $\phi \geq N\theta$. We define the sensing range of a monocular camera s_{M} the maximum distance at which the obstacle is still detectable, namely the distance at which the previous condition is satisfied with the equality constraint:

$$s_{\text{M}} = \frac{r_o}{2\tan\left(\frac{N\theta}{2}\right)}. \tag{C.10}$$

Eq. (C.10) shows that the sensing range of a monocular camera depends on its angular resolution θ. Fig. C.7 shows the range at which a monocular system can detect an obstacle of size $r_o = 0.5\,\text{m}$ when this occupies a percentage $k = 5\%$, $k = 10\%$ and $k = 15\%$ of the image size q.

C.9.2 Latency

The latency of a camera-based perception system depends on (i) the time t_{f} between two consecutive images, (ii) the number of images necessary for detection, and (iii) the time to process each image. The first one only depends on the sensor itself, and includes, among the other things, the exposure time and the transfer time. The second and the third depend on the sensor, the computational power available and the algorithm used to detect the obstacle. It is therefore hard to provide an exact estimate of the actual latency of a perception system based on a monocular camera, since it depends on a

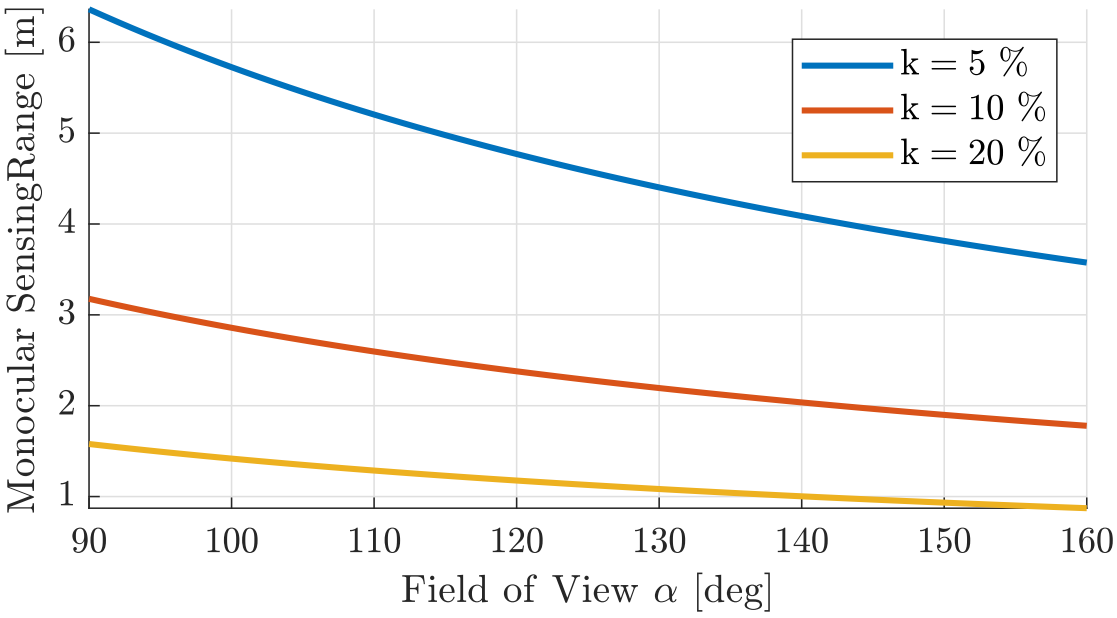


Figure C.7: The sensing range s_M for a monocular system depending on the field of view α. The the number of pixels N necessary to detect an obstacle of size $r_o = 0.5\,\text{m}$ are computed as a percentage k of the image resolution.

large variety of factors. Thus, in this work we analyze its theoretical upper-bound and lower-bound to provide a *back-of-the-envelope* analysis of the possible performance achievable.

For a vision-based perception system to be effective, it has to produce its output in real-time. This means that, if n is the number of images necessary for the detection, the latter must happen before the frame $n+1$ arrives. Therefore, the frame-rate t_f of a camera provides an upper-bound for the latency of a monocular vision system. Assuming that 2 frames are sufficient to detect an obstacle along the robot's path, the latency for a monocular camera has an upper-bound given by $\tau_M = 2t_f$.

To have an estimate of the theoretical lower-bound of the latency of a frame-based camera, we neglect the processing time and only consider the delays caused by how such cameras work. More specifically, in the ideal case of negligible processing time, the lower-bound of the latency depends on (i) the time t_f between two consecutive triggers of the sensor, (ii) the exposure time t_E, and (iii) the time t_T necessary to transfer each frame. In the ideal case of no processing time, the latency of a frame-based camera has a lower-bound $\tau_M = t_f + t_T + t_E$. Typically, an image is transferred to the processing unit before the next one arrives, which means $0 < t_T < t_f$. The time t_T depends on the size of the image and the protocol used to communicate with the sensor. For example, a gray-scale VGA resolution image (i.e., $640 \times 480\,\text{pxl}$) has a size of 2.1 Mbit and can be transferred in approximately 5 ms with a USB 2.0 connection (480 Mbit/s) and 0.4 ms

with a USB 3.0 connection (5 Gbit/s). The exposure time depends on the amount of light available in the environment and cannot be larger than the time between two consecutive frames, i.e. $0 < t_E < t_f$.

C.10 Stereo Frame-Based Camera

C.10.1 Sensing Range

Using stereo cameras, it is possible to triangulate points using only one measurement consisting of two frames grabbed at the same time. Let b be the baseline between the two cameras, f their focal length and l the disparity between the two images of a point of interest. The depth of such a point is given by $d = f\frac{b}{l}$. However, the uncertainty in the depth estimation ϵ_D grows proportional to the square of the distance between the camera and the scene [62], namely $\epsilon_D = \frac{z^2}{bf}\epsilon_P$, where ϵ_P is the uncertainty in the disparity matching. Therefore, we consider the sensing range for a stereo camera s_S as the maximum depth such that the uncertainty in the depth estimation is below a given percentage threshold k:

$$s_S = \frac{kfb}{\epsilon_P}. \tag{C.11}$$

Fig. C.8 shows the sensing range of a stereo camera as a function of the baseline b such that the depth uncertainty ϵ_D is below 5% and 20% of the actual depth, for the cases of VGA (640×480 pxl) and QVGA (320×240 pxl) resolutions, assuming $\epsilon_P = 1$ pxl.

C.10.2 Latency

Differently from monocular systems, stereo cameras capture simultaneously two frames using two cameras placed at a relative distance b. It is therefore possible to use a single measurement, i.e. two frames from two different cameras, to detect obstacles, for example computing the disparity between such frames, a depth map or an occupancy map. Depending on the technique used to detect obstacle using a stereo camera, the computational power available and the resolution of the output, the latency of a stereo system can vary significantly. For example, the Intel RealSense, provides a depth map at a frequency of 60 Hz (RealSense R200[2]), while the Bumblebee XB3[3] only provides its output at up to 16 Hz. However, computing the latency of those measurements is not an easy task, since most of the commercially available sensors do not provide such information in their datasheets. An estimate of the latency of a wide

[2] https://tinyurl.com/realsenser200
[3] https://tinyurl.com/bumblebeexb3

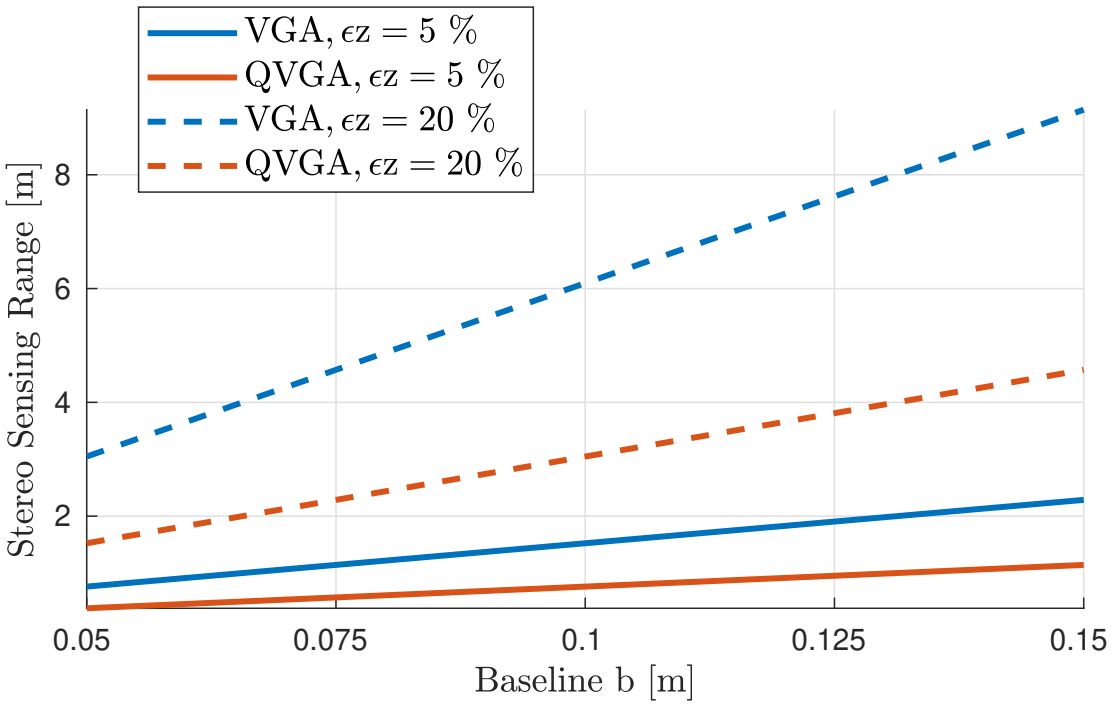


Figure C.8: The sensing range s_S for a stereo system depending on the baseline b for a focal length of 4 mm. This sensing range guarantees that the uncertainty ϵ_D is below 5% and 20% of the actual depth.

variety of depth cameras is available thanks to the effort of the robotics community[4], according to which most of the stereo systems have a latency of one frame. Therefore, we consider as a lower-bound for stereo cameras the inverse of the frame-rate of the fastest sensor currently available on the market, namely the Intel RealSense, leading to a lower-bound $\tau_S = 0.017\,\text{s}$. For the upper-bound, instead, we can refer to the datasheet of the Stereolab ZED Mini[5], which has an estimated latency $\tau_S = 0.07\,\text{s}$.

C.11 Monocular Event Camera

C.11.1 Sensing Range

Since monocular frame-based cameras and event camera often share the same sensor, we can use (C.10) to compute the sensing range of an event camera. However, the amount of pixels the obstacle must occupy in the image in order to be detected is significantly smaller. In principle, the obstacle would generate an event when each of its two edges occupy at least 1 pxl in the image. However, due to the noise of this sensor, the obstacle can be detected with an event camera when it occupies an amount of pixels in the image which is significantly larger than the amount of pixels it has to move to generate an event (see Sec. C.11.2 of this document). In this work, we assume that the obstacle size in the image must be at least one order of magnitude larger than the amount of pixels it has to move to fire an event. Therefore, we compute the sensing range of an event camera using (C.10) with $N = 10$. This leads to a sensing range for an event camera which, depending on the field of view of the sensor, can span between $s_E = 10\,\text{m}$ and $s_E = 20\,\text{m}$ for an obstacle of size $r_o = 0.5\,\text{m}$.

C.11.2 Latency

In this work, we assume that an obstacle can be detected using an event camera whenever its edges generate an event. For this to happen, there must be sufficient relative motion between the camera and the obstacle to cause a change of intensity sufficiently large to let an event fire. Typically, as shown in [125], the edge of an obstacle generates an event when its projection on the image plane moves by at least 1 pxl. Without loss of generality, we analyze the horizontal motion of the obstacle in the image. Let d be the distance between the robot and the obstacle along the camera optical axis, and let r_o be the radius of the obstacle. Furthermore, assume the optical axis of the camera to pass through the geometric center of the obstacle, which we model here as a segment (cf. Fig. C.9). The projection of a point p into the image plane has

[4] https://rosindustrial.org/3d-camera-survey/
[5] https://www.stereolabs.com/zed-mini/

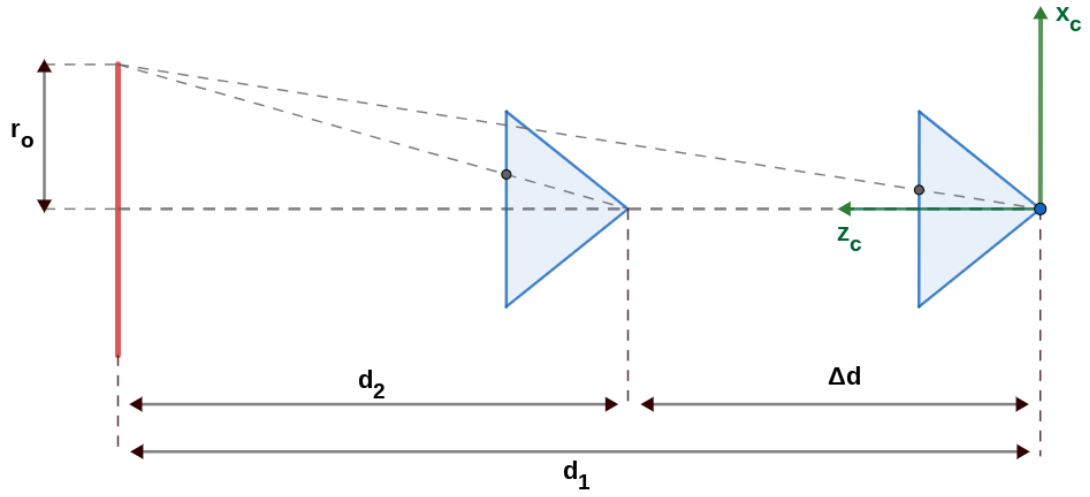


Figure C.9: A schematics representing the translation necessary for the obstacle to generate an event. The obstacle (in red on the left side of the picture) is projected on the image plane on a point which has horizontal component u depending on the distance d_i from the camera. The quantity Δd represents the distance the camera has to move such that the projection edge of the obstacle on the image plane moves by 1 pxl.

horizontal component u given by [70]:

$$u = f\frac{{}_cp_x}{{}_cp_z}, \tag{C.12}$$

where ${}_cp_x$ and ${}_cp_z$ are the components of p in the camera reference frame, and f is the camera focal length. In our case, ${}_cp_x = r_o$ and ${}_cp_z = d$. We can compute (C.12) for two values d_1 and $d_2 = d_1 - \Delta d$ of the distance along the optical axis, obtaining two different values u_1 and u_2, respectively. Equating $\Delta u = u_2 - u_1$ to the desired translation in the image plane necessary to generate an event (in our case, $\Delta u = 1$ pxl), we can compute the camera translation Δd as:

$$\Delta d = \frac{\Delta u d_1^2}{f r_o + \Delta u d_1}. \tag{C.13}$$

The time it takes the robot to cover such a distance Δd depends on its speed $\hat{v}_1$:

$$\tau_E = \frac{1}{\hat{v}_1}\frac{\Delta u d_1^2}{f r_o + \Delta u d_1}. \tag{C.14}$$

Eq. (C.14) shows the time necessary to get an event from the edge of the aforementioned obstacle.

It is important to note that, since the transfer time for an event is in the order of a few microseconds [126], we consider it negligible. Similarly, we neglect the processing time

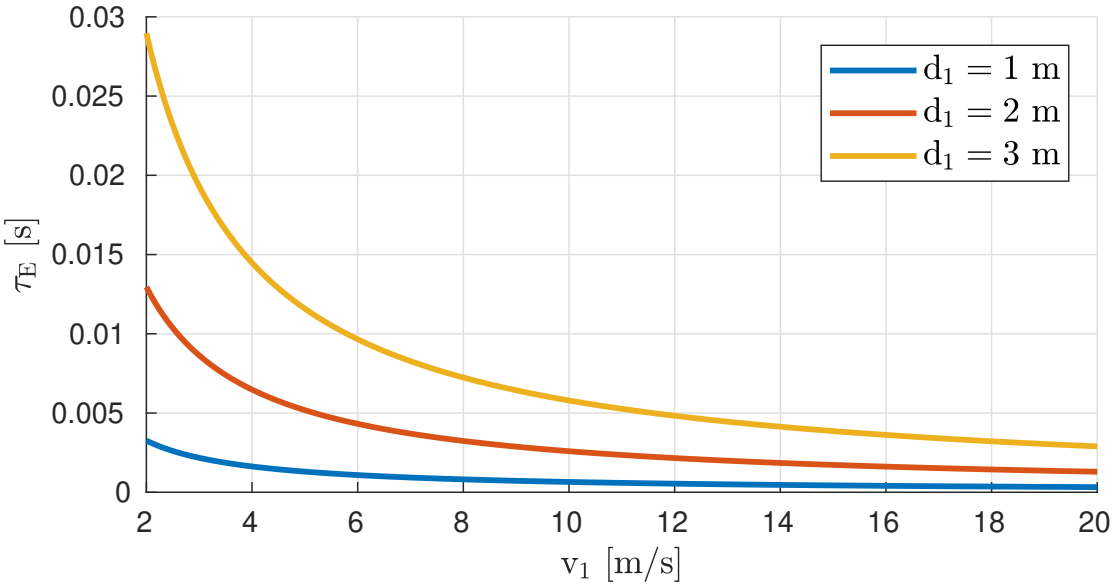


Figure C.10: The latency τ_E for an event camera depending on its distance from the obstacle d and the speed $\hat{v}_1$. We considered the case of an event camera with a VGA resolution sensor and a focal length of 4 mm.

for the case of event-based vision, since each pixel triggers asynchronously from the other and, therefore, the amount of data to be processed is significantly lower than the case where an entire frame has to be analyzed.

Fig. C.10 shows the latency for an event camera (C.14) depending on its distance from the obstacle d and the speed $\hat{v}_1$ in the case of VGA resolution and focal length of 4 mm. It is clear that the theoretical latency of an event camera is not constant, but rather depends on the relative distance and the speed between the camera and the obstacle. Therefore, to compute the maximum latency that a robot can tolerate in order to safely navigate using an event camera, it is necessary to jointly consider the sensing range (Sec. C.3.2), and Eq. (C.8) and (C.14). Intuitively speaking, this is due to the fact that event cameras are motion activated sensors. In order for the edges of an obstacle to generate an event, their projection in the image must move by at least 1 pxl. For this to happen, the robot must move towards the obstacle by a quantity Δd which depends on its distance to the obstacle through (C.13). Therefore, the latency of an event camera, i.e. the time it takes the obstacle to generate an event, is given by the ratio between such a distance Δd and the robot's speed $\hat{v}_1$, as shown by (C.14). However, the relative distance and speed between the robot and the obstacle also influence the time to contact (C.3), which must be larger than the avoidance time (C.4) for the robot the able to avoid the obstacle before colliding with it. Therefore, it is not possible to arbitrarly reduce the latency of an event camera for obstacle avoidance by increasing the robot's speed, since this might result in unfeasible avoidance maneuvers.

C.12 Discussion

C.12.1 Stereo Frame or Monocular Event?

As shown in Tab. C.1, stereo cameras and event cameras provide results that, at least for currently available quadrotors, are comparable in terms of magnitude. Stereo cameras are currently still among the best options for autonomous quadrotor flight, since they provide a good compromise between latency and sensing range, without being very expensive. However, technological development in the event cameras might render them better solutions in the the future since (i) increasing the resolution would lead to higher angular resolution, which results in longer ranges, and (ii) they will be become cheaper as mass-production starts. Also, the sensing range of stereo cameras strongly depends on the baseline between the two cameras, which for small quadrotors are not always possible. Additionally, carrying one camera instead of two makes the platform lighter and, therefore, more agile [90]. Finally, event cameras have other advantages compared to frame-based cameras such as: (i) high dynamic range, which makes them more suited for navigation in adverse lighting conditions, where frame-based cameras might fail; (ii) their latency does not depend on the exposure time, which plays an important role in frame-based cameras and can significantly increase their latency; (iii) high temporal resolution, which reduces the motion blur and makes obstacle detection easier at high speed; (iv) low power consumption, which is desirable with small-scale robots [140].

C.12.2 Dynamic Obstacles

In this work, we only considered the case of navigation through static obstacles. Nevertheless, the mathematical framework provided in Sec. C.2 can be used to consider the case of moving obstacles by taking into account that, in that case, the time to contact and the avoidance time depend on the relative distance and speed between the robot and the obstacle along the longitudinal and the lateral axes.

A fundamental assumption of our work is that the robot moves along a direction which makes the obstacle detectable and eventually leads to a collision. In the case of moving obstacles, this might not always be the case. Indeed, depending on the relative distance and speed between the robot and the obstacle, a number of cases can occur: (i) the robot detects the obstacle, but their relative motion does not lead to a collision; (ii) the robot detects the obstacle, and their relative motion leads to a collision; (iii) the robot cannot detect the obstacle, and their relative motion does not lead to a collision; (iv) the robot cannot detect the obstacle, but their relative motion leads to a collision. It is clear that, in the case of moving obstacle, the amount of cases to be taken into account and the parameters to be considered increases significantly. For example, the field of view of the robot also plays a crucial role in the case of moving obstacles. Indeed,

for a given relative speed, depending on the field of view of the sensing pipeline it is equipped with, the robot might or might not be able to detect the obstacle. In the case it is able to detect the obstacle, the relative distance at the moment the latter enters the sensing range depends on how large the field of view is, which then determines the time to contact. Therefore, in the case of a robot navigating through moving obstacles, a broader and more detailed analysis of the dependence of the maximum achievable speed on each parameter is necessary.

Intuitively, moving obstacles would highlight the benefits of event cameras against other sensors. To compute the latency of an event camera, we considered the case of a robot moving towards a static obstacle, placed in the center of the image, along a direction parallel to the camera's optical axis. This represents a sort of *worst case* for event cameras, since the apparent motion between the sensor and the obstacle is small. Conversely, an obstacle moving along the lateral axis would increase the apparent motion in the image and, therefore, generate an event earlier than in the case of static obstacles. Additionally, when obstacles enter the sensing area at a short distance, the importance of latency increases as the time to contact decreases. For this reason, we expect that event cameras would allow faster flight in the case of moving obstacles, especially for short sensing ranges (or, equivalently, for obstacles entering the sensing area at short distances). We are currently working on analyzing the impact of the sensing pipeline's parameters (latency, sensing range and field of view) for the case of moving obstacles from a mathematical point of view.

C.13 Experiments

C.13.1 Experimental Platform

We used a custom-made quadrotor platform to perform the experiments. The vehicle was built using the DJI F330 frame, and was equipped with Cobra CM2208 motors and Dalprop 6045 propellers. The tip-to-tip diagonal of the quadrotor was 50 cm, with an overall take-off weight of approximately 860 g and a thrust-to-weight ratio of roughly 3.5. We used an Optitrack motion-capture system to measure the state of the quadrotor, as well as the position and velocity of the ball. The ball measurements were not used by the vehicle, which only relied on the information coming from the onboard obstacle detection algorithm, and were used as ground truth to benchmark the sensing pipeline. To detect the obstacle, we mounted an Insightness SEEM1 [6] neuromorphic sensor looking forward, and an Intel UpBoard computer running the obstacle detection algorithm described in the previous section. The horizontal field of view of the sensor was approximately 90°. Whenever the obstacle was detected, a trigger signal was sent to a ground-station computer connected to the motion-capture system and running

[6] http://www.insightness.com/technology

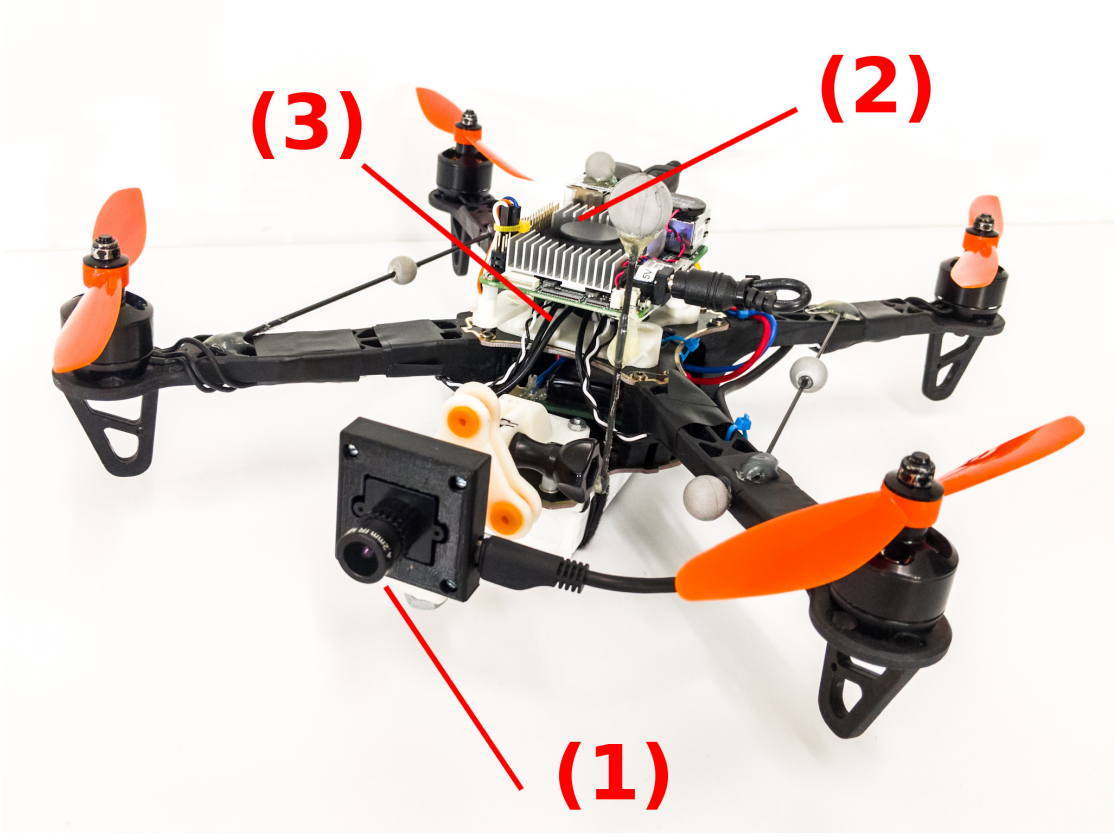


Figure C.11: The quadrotor used for the experiments. (1) The Insigthness SEEM1 sensor. (2) The Intel Upboard computer, running the detection algorithm. (3) The Lumenier F4 AIO flight controller, receiving commands from the ground station.

the control stack described in [44], which then initiated the avoidance maneuver. The control commands (i.e., collective thrust and body rates) were sent to a Lumenier F4 AIO flight controller by the ground-station through a Laird RM024 radio module.

C.13.2 Obstacle Detection with an Event Camera: Theoretical and Practical Latency

As described in Sec. C.5 of the main manuscript, we performed actual experiment on a quadrotor equipped with an Insightness SEEM1 sensor having QVGA resolution (i.e., 320×240 pxl). We estimated that, in order to obtain reliable measurements of the obstacle, a displacement Δu of 5 pxl was typically necessary. Fig. C.12 shows the theoretical latency of such a sensor for obstacle detection, according to the model proposed in Sec. C.11.2, for a sensing range of 1 m, 2 m and 3 m, with $\Delta u = 5$ pxl.

To validate these results, we performed a quantitative analysis using ground truth data provided by an Optitrack motion-capture system. More specifically, we performed 100 experiments throwing the ball, anchored to a table through a leash to prevent collisions, towards the quadrotor, and used data from the motion-capture system to measure the moment when the ball entered the sensing range s of the camera. This was manually set to three different values, i.e. $s = 1$ m, $s = 2$ m and $s = 3$ m. For each of these values, we computed the time when the sensing pipeline detected the ball for the first time, and compared it to the time when the obstacle actually entered the sensing range using data from the motion-capture system. This comparison allowed us to estimate the latency of our event-based obstacle detection algorithm, and the results are shown in Fig. C.13 for a range of obstacle speeds between $5\,\mathrm{m\,s^{-1}}$ and $9\,\mathrm{m\,s^{-1}}$.

s [m]	μ [s]	σ [s]
1	0.0037	0.0030
2	0.0688	0.0474
3	0.1832	0.0766

Table C.2: The mean μ and standard deviation σ of the latency for the obstacle detection algorithm proposed in this work based on the Insightness SEEM1 sensor.

C.13.3 Obstacle Detection with an Event Camera: Discrepancy Between Theory and Practice

As one can notice, the results in Fig. C.13 agree with the theoretical lower-bound of the latency expected for the sensor used in our experiments, shown in Fig. C.12. Tab. C.2 reports the mean μ and standard deviation σ of the latency of our event-based obstacle detection algorithm, depending on the desired sensing range. As the sensing range increases, also the error between the mean value and the expected theoretical latency

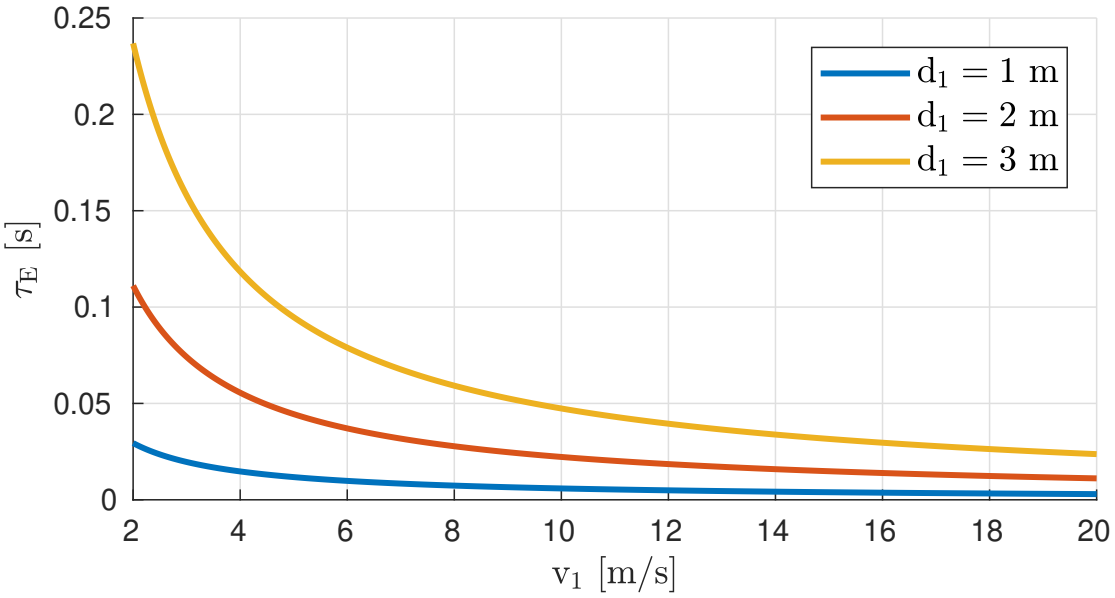


Figure C.12: The theoretical latency τ_E for the Insigthness SEEM1 used in our experiments, depending on its distance from the obstacle d and the speed $\hat{v}_1$. We considered the case of an event camera with a QVGA resolution sensor and a focal length of 4 mm.

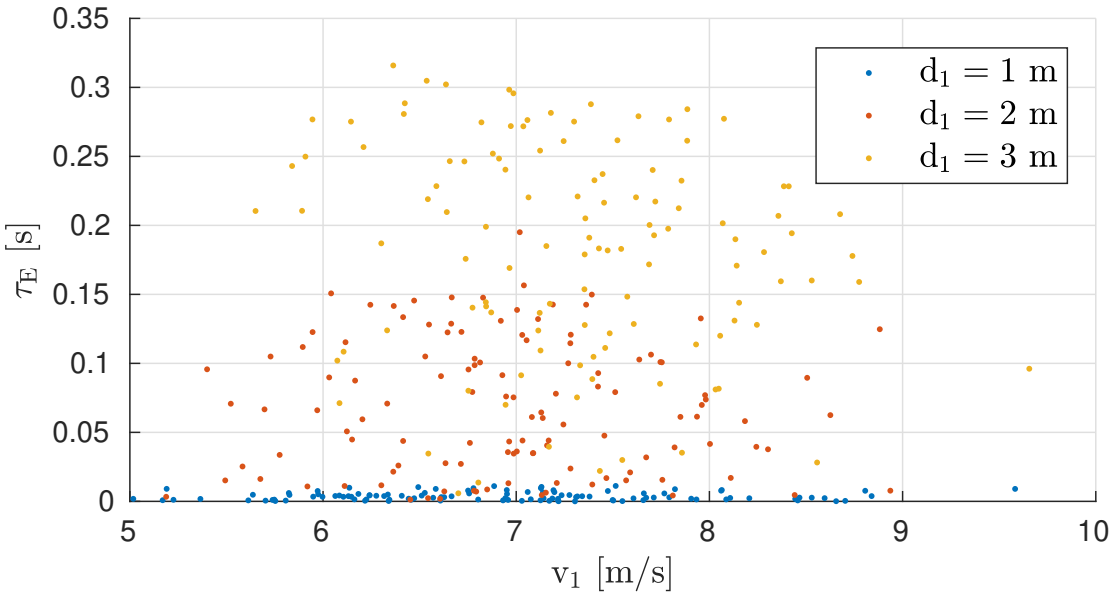


Figure C.13: The measured latency τ_E of our event-based obstacle detection algorithm using an Insightness SEEM1, depending on its distance from the obstacle d and the speed $\hat{v}_1$.

increases. Similarly, the standard deviation becomes larger. We believe this effects to be mainly due to two factors.

First, the output of an event camera is particularly noisy. The higher the noise level, the larger the amount of events that need to be processed and, therefore, the higher the computational cost of our algorithm. In our case, the noise comes from both actual sensor noise and events generated by the static part of the scene which are not perfectly compensated by our algorithm.

Second, the resolution of our sensor is particularly low. This has a twofold consequence. The first is that the size of the obstacle in the image is not very large when it is far away from the camera. The second is that, when the obstacle is far from the camera, it needs to move by a significant amount in order for its projection in the image to move by an amount $\Delta u = 5\,\text{pxl}$. The closer it gets to the camera, the smaller the distance it has to travel to produce such displacement Δu, which, for a constant velocity of the obstacle, translates into a lower detection latency. Additionally, when the obstacle is close to the camera, it occupies a significant portion of the image, making its detection easier.

Therefore, as the sensing range increases, the difference between the theoretical model (Sec. C.11.2) and the actual sensing pipeline becomes more and more important. However, near-future improved versions of event-based sensors can bridge this gap and render event-based obstacle detection pipelines closer to the theoretical model we propose in this work. More specifically, we believe that event cameras with higher resolution could lead to better and faster obstacle detection pipelines. An additional benefit of large resolutions is the possibility of mounting lenses providing larger field of views, which are desirable to sense obstacles, without sacrificing the angular resolution of the sensor.

D Event-Based Avoidance

D. Falanga, K. Kleber, and D. Scaramuzza. "Low Latency Avoidance of Dynamic Obstacles for Quadrotors with Event Cameras". In: *AAAS Science Robotics, Under Review* (2019)

Low Latency Avoidance of Dynamic Obstacles for Quadrotors with Event Cameras

Davide Falanga, Kevin Kleber, and Davide Scaramuzza

Abstract — In this paper, we address one of the fundamental challenges for micro aerial vehicles: dodging fast moving objects using only onboard sensing and computation. Effective avoidance of moving obstacles requires fast reaction times, which entails low-latency sensors and algorithms for perception and decision making. All existing works rely on standard cameras, which have latencies of tens of milliseconds and suffer from motion blur. We depart from state of the art by relying on a novel bioinspired sensor, called event camera, with reaction times of microseconds, which perfectly fits our task requirements. However, because the output of this sensor is not images but a stream of asynchronous events that encode per-pixel intensity changes, standard vision algorithms cannot be applied. Thus, a paradigm shift is necessary to unlock the full potential of event cameras. Our proposed framework exploits the temporal information contained in the event stream to distinguish between static and dynamic objects, and makes use of a fast strategy to generate the motor commands necessary to avoid the detected obstacles. Our resulting algorithm has an overall latency of only 3.5 ms, which is sufficient for reliable detection and avoidance of fast-moving obstacles. We demonstrate the effectiveness of our approach on an autonomous quadrotor avoiding multiple obstacles of different sizes and shapes, at relative speeds up to $10\,\mathrm{m\,s^{-1}}$, both indoors and outdoors.

Figure D.1: Sequence of an avoidance maneuver.

Videos of the Experiments

All the experiments reported in this manuscript is available at http://rpg.ifi.uzh.ch/event_based_avoidance

D.1 Introduction

Micro aerial vehicles (MAVs) are at the forefront of this century's technological shift. They are becoming ubiquitous, giving birth to new, potentially disruptive markets worth several billion dollars, such as aerial imaging (forecast value of 4 billion USD by 2025 [1]), last-mile delivery (90 billion USD by 2030[2]) and aerial mobility (almost 8 billion USD in 2030 [3]).

Keeping a vehicle airborne above a crowd poses large safety risks. Several drone crashes have been recently reported in the news, due to either objects tossed at quadrotors during public events [4] [5], or collisions with birds [6] [7]. Enabling MAVs to evade fast-moving objects (cf. Fig. D.1) is therefore critical for the deployment of safe flying robots on a large scale and is still unsolved.

D.1.1 The Challenge

The temporal latency between perception and action plays a key role in obstacle avoidance. The higher the latency, the lower the time the robot has to react and execute an avoidance maneuver [47]. This is especially critical for MAVs, where a collision can not only damage the environment, but also cause severe hardware failure. Additionally, micro quadrotors have reduced payload capabilities, which puts a hard bound on the

[1] http://bit.do/aerial-imaging-market
[2] http://bit.do/aerial-delivery-market
[3] http://bit.do/air-mobility-market
[4] http://bit.do/forbes-drone-taken-down
[5] http://bit.do/standard-argentina-drone-takedown
[6] http://bit.do/daily-mail-eagle-drone
[7] http://bit.do/cnet-hawk-drone

sensing and computing resources they can carry.

The existing literature on obstacle avoidance for MAVs relies on standard cameras (in a monocular [38, 3, 97] or stereo configuration [138, 20, 121, 10]) or on depth cameras [98, 100, 77]. However, these works assume that the obstacles in the environment are either static or quasi-static (i.e., slow relative motion).

Similarly, state-of-the-art consumer drones are not currently capable to reliably detect and avoid moving obstacles. For example, the Skydio drone, as of today one of the most advanced autonomous drones on the market, is not capable of dealing with moving objects (*If you throw a ball at it, it's almost certainly not going to get out of the way*, said Adam Bry, CEO of Skydio [8]).

Developing effective algorithms to avoid dynamic obstacles is therefore a key challenge in robotics research, as well as a highly admired goal by major industry players.

D.1.2 Event Cameras

The limitations of standard cameras arise from their physical nature and cannot be solved with sophisticated algorithms. The solution is given by a novel type of sensor, called event camera, which has a sensing latency virtually negligible.

Event cameras [96] are bio-inspired sensors that work radically different from traditional cameras. Instead of capturing images at a fixed rate, an event camera measures per-pixel brightness changes asynchronously. This results in a stream of events at microsecond resolution. More specifically, an event camera has smart pixels that trigger information *independently* of each other: whenever a pixel detects a change of intensity in the scene (e.g., caused by relative motion), that pixel will trigger an information at the time the intensity change was detected. This information is called *event*, and encodes the time (at microsecond resolution) at which the event occurred, the pixel location, and the sign of the intensity changes. Let $t_{k\text{-}1}$ be the last time when an event fired at a pixel location $\mathbf{x}$, and let $L_{k\text{-}1} = L(\mathbf{x}, t_{k\text{-}1})$ be the intensity level at such pixel at time $t_{k\text{-}1}$. A new event is fired at the same pixel location at time t_k as soon as the difference between the intensity $L_{k\text{-}1}$ and L_k is larger than a user-define threshold $C > 0$. In other words, an event is fired if $\|L(\mathbf{x}, t_k) - L(\mathbf{x}, t_{k\text{-}1})\| > C$ (positive event) or $\|L(\mathbf{x}, t_k) - L(\mathbf{x}, t_{k\text{-}1})\| < -C$ (negative event). We refer the reader to [61] for further details.

To better highlight what happens across the entire sensor, we compare the output of an event camera to the one of a conventional camera in Fig. D.2 and in a video [9].

Event cameras can thus be seen as *asynchronous*, *motion-activated* sensors, since they

[8] http://spectrum.ieee.org/automaton/robotics/drones/skydio-r1-drone
[9] https://youtu.be/LauQ6LWTkxM?t=32

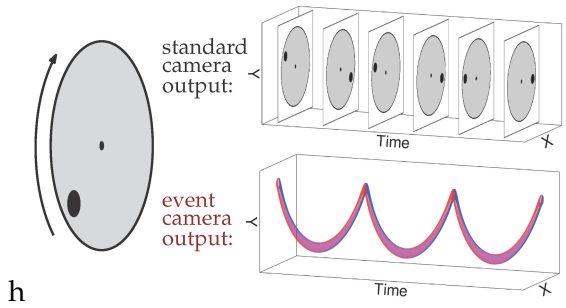


Figure D.2: Comparison of the output of a conventional camera and that of an event camera when they are looking at a rotating disk with a black dot. While a conventional camera captures frames at a fixed rate, an event camera only outputs the sign of brightness changes continuously in the form of a spiral of events in space-time (red: positive changes, blue: negative changes).

provide measurements only *if* and *where* there is relative motion between the camera and the environment. And because their latency is in the order of microseconds, they are a natural choice for detection and avoidance of fast moving obstacles by flying MAVs.

If one removes the events induced by the ego-motion of the vehicle [119, 178], one can directly obtain information about the moving part of the scene. This leads to multiple advantages over standard cameras for detection of dynamic obstacles: (i) the output is sparser and lighter than a frame, therefore cheaper to process; (ii) no segmentation between static and dynamic objects is necessary, since to do so it is possible to exploit the temporal statistic of each event; (iii) their high temporal resolution (in the order of microseconds) allows low-latency sensing.

D.1.3 Related Work

In recent years, event cameras have attracted the interest of the robotics community [61]. Obstacle detection is among the applications with the highest potential, and previous works investigated the use of these sensors to detect collisions [161] and track objects [119, 113]. However, very few examples of closed-loop control based on event cameras are available in the literature. Among these, the majority of the works focuses on simple, low-dimensional tasks, such as 1-DoF heading regulation [21, 127], stereo-camera gaze control [65, 66], 2-DoF pole balancing [27], 1-DoF robotic goalkeeping [33, 32], or navigating ground robots among static obstacles [25, 63, 13].

Examples of closed-loop control of more complex robotic systems, such as quadrotors, using event cameras are our recent works [154, 126, 162]. In [154], we proposed an event-based visual-inertial odometry algorithm for state estimation and closed-loop trajectory tracking of a quadrotor. Instead, [126] and [162] are the most related to this paper. In [126], we analyzed the feasibility of detecting spherical objects thrown at a

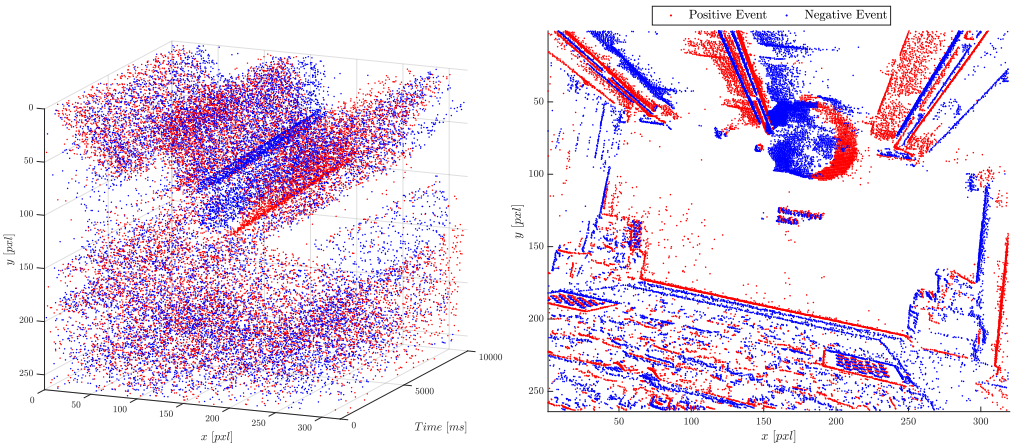
stationary event camera on small embedded processors for quadrotors. In [162], we showed preliminary results in using shallow neural networks for segmenting moving objects from event streams and demonstrated an application to quadrotor obstacle avoidance. However, the resulting sensing latency was 60 ms rather than the 3.5 ms of this paper, thus strongly limiting the maximum relative speed at which moving objects could be evaded. Additionally, differently from this paper, there we did not consider the relative distance and velocity to compute the avoidance commands. To the best of our knowledge, this is the first work that implements and demonstrates low latency (3.5 ms) dynamic obstacle dodging on an autonomous quadrotor with relative speeds up to $10\,\mathrm{m\,s^{-1}}$.

D.1.4 Overview of the Approach and Contributions

Our moving-obstacle detection algorithm works by collecting events during a short-time sliding window and compensating for the motion of the robot within such time window. Fig. D.3 shows the effects of the ego-motion compensation: on the left side, the 3D volume of the events accumulated during an arbitrary time window of 10 ms, on the right side the same events, after ego-motion compensation, back-projected on the image plane. We analyze the temporal statistics of the motion-compensated events to remove those generated by the static part of the environment.

Broadly speaking, our algorithm is based on the intuition that the the static part of the scene fires events uniformly across the entire time window and, after the ego-motion compensation, the pixels where they belong show a uniform distribution of timestamps; conversely, dynamic objects generate ego-motion compensated events that are accumulated around specific sections of the time window, and can therefore be distinguished. An intuitive explanation of how and why our algorithm works is provided in Sec. D.1.5, while we refer the reader to Sec. D.3.1 for a detailed explanation of this process, which allows us to obtain a so-called *event frame*, containing only events coming from moving objects, at very high rate. We leverage a fast clustering algorithm to tell apart different objects in the event frame, and use a Kalman filter to obtain information about their velocity. Fig. D.4 provides a visual explanation of the steps involved in our algorithm, which is thoroughly described in Sec. D.3.1.

The position and velocity of each obstacle relative to the camera is then fed to a fast avoidance algorithm designed to leverage the low sensing latency. To do so, we use a reactive avoidance scheme based on the use of artificial potential fields [85] relying on fast geometric primitives to represent the obstacles, which renders it computationally inexpensive. We propose a novel formulation of the repulsive field which better suits the task of avoiding fast moving obstacles by taking into account the need for a prompt reaction of the robot when an obstacle is detected. Compared to previous approaches, our formulation of the repulsive potential increases significantly faster as the distance



(a) The 3D volume of the events generated within a time window of 10 ms.

(b) The same events, projected into the image plane after ego-motion compensation.

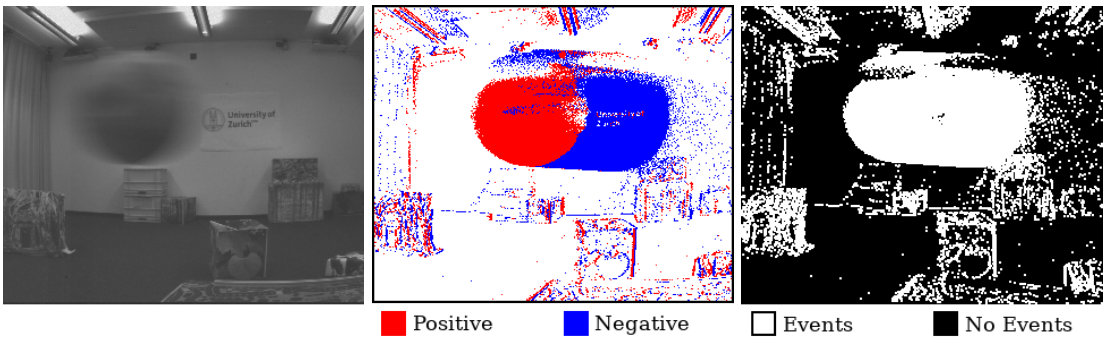
Figure D.3: Our algorithm collects all the events that fired during the last 10 ms, here represented in the 3D volume on the left side, and used the Inertial Measurement Unit to compensate for the motion of the camera. The ego-motion compensated events are therefore projected into a common image frame, here shown on the right side, where each pixel contains potentially multiple events. By analyzing the temporal statistics of all the events projected into each pixel, our approach is able to distinguish between pixels belonging to the static part of the scene and to moving objects.

between the robot and the obstacle decreases in order to render the avoidance maneuver more reactive and agile. Additionally, we consider both the magnitude and the direction of the obstacle's velocity to decide in which direction to evade, and introduce a decay factor in the magnitude of the potential to take into account that the obstacles we consider are dynamic, i.e. they do not occupy the same position in time. Further details about the avoidance strategy are available in Sec. D.3.2.

Our approach prioritizes computation speed over accuracy, therefore, we trade-off detection accuracy for latency. Nevertheless, in Sec. D.2.1 we show that our algorithm only takes on average 3.5 ms (from the moment it receives the events to process to when it sends the first command to avoid the detected obstacles) to detect moving obstacles with a position error usually in the order of a few tens of centimeters.

Trading-off detection accuracy for latency is not only a necessity for robotic platforms, but it has been frequently observed also among animals [23] for the execution of several tasks involving visual sensing.

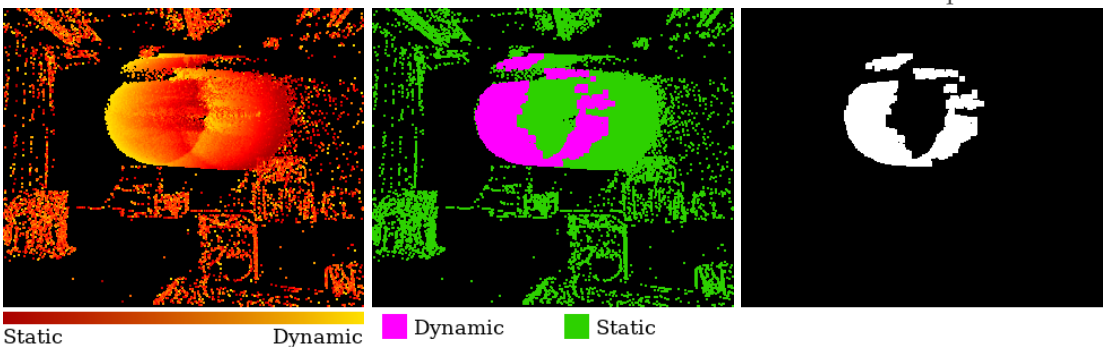
We demonstrate the effectiveness of our approach in real experiments with a quadrotor platform. We validate our system with both a monocular setup (for obstacles of known size) and a stereo setup (for obstacles of unknown size), both indoors and outdoors. The entire avoidance framework is capable of running in real-time on a small single-board computer on-board the vehicle, together with the entire software stack necessary to let



(a) Frame provided by the Insightness SEEM1, which shows clearly the motion blur due to the relative motion between the sensor and the moving obstacle.

(b) All the events accumulated in a time window. Red indicates positive events, while negative events are reported in blue.

(c) The same events, motion-compensated using the IMU: if a pixel is colored in white, it means that at least one event has been backprojected there after motion-compensation.



(d) The motion-compensated events, with color code representing the normalized mean timestamp (Eq. (D.4)): events belonging to the dynamic part of the scene are represented in yellow.

(e) Mean-timestamp image after threhsolding: green and purple indicate the static and the moving part of the scene, respectively.

(f) Events belonging to moving obstacles. This frame is used to segment out the different dynamic objects in the scene (Sec. D.3.1).

Figure D.4: A figure summarizing all the steps of our ego-motion compensation algorithm to isolate the events belonging to moving obstacles. Fig. D.4a shows a frame capture by the Insightnes SEES1 camera. Fig. D.4b reports all the events accumulated in the last window, with red and blue indicating the polarity (positive and negative, respectively). Fig. D.4c reports the result of the ego-motion compensation, showing in white all the pixels where there has been at least one event in the time window. We compute the normalized mean timestamp of all the events belonging to a given pixel, and the resulting values are shown in Fig. D.4d. Based on the normalized mean timestamp, we can disambiguate between the events belonging to the static part of the scene and those belonging to the dynamic objects (Fig. D.4e, where green represents static events and purple moving events). Finally, we obtain a frame containing only the events belonging to the dynamic part of the scene, as shown in Fig. D.4f.

the robot fly (i.e., state estimation, high level control, communication with the flight controller). Experimental results show that our framework allows the quadrotor to avoid obstacles moving towards it at relative speeds up to $10\,\mathrm{m\,s^{-1}}$ from a distance of around 3 m.

D.1.5 Time Statistics of Events to Detect Moving Obstacles

To provide an intuitive example of how and why our algorithm successfully classifies static and dynamic events, Fig. D.5 shows the simplified case of a mono-dimensional event camera (i.e., an event camera having only one row), rotating in a plane while observing both a static and a dynamic object. The dynamic object (in red) moves from left to right, while the event camera rotates counter-clockwise.

In the center of the figure, we consider a time window spanning from an initial time t_1 to a final time t_5, and we discretize this interval into five time instants to visualize the sequence of events generated by both the motion of the camera and the dynamic object. Let us assume that at time t_1 both objects generate an event due to the motion of the camera: the static object fires an event at pixel p_1, the dynamic object at pixel p_2. At time t_2, the motion of the dynamic object causes another event at pixel p_3, while at time t_3 the motion of the camera generates events at pixels p_2 (static) and p_4 (dynamic). The same concept applies to times t_4 and t_5. After collecting all these events, if we motion-compensate them to remove the effects of the motion of the camera, we obtain a situation like the one depicted at the bottom of the center part of the image, where multiple events get back-projected into the same pixel location.

On the right side of the figure, we report the time statistics of the event project into pixels p_1 to p_4, which are the only ones having motion-compensated events. As one can see, the events belonging to the static part of the scene are equally spread across the time window, while the events fired due to the motion of the dynamic object are concentrated either at the beginning, the center, or the end of the window. If we now compute the mean timestamp of all the events falling in each pixel, subtract the mean of all the events and normalize it by the length of the time window, we obtain a score for each pixel spanning between -1 and 1. We expect events belonging to the static part of the scene to have a score of approximately zero, since they contain events spread across the entire window more or less uniformly. On the contrary, events belonging to the dynamic part of the scene have scores that can span between -1 and 1, depending on where they are concentrated within the time window. In particular, the events generated by the dynamic object at the beginning of the window have a score of -1, those fired at the center of the window have a zero score, while those generated at the end of the window have a score of approximately 1. Since we are interested only about the latest position of the dynamic obstacles, we discard non-positive scores, taking into account only events with score above zero.

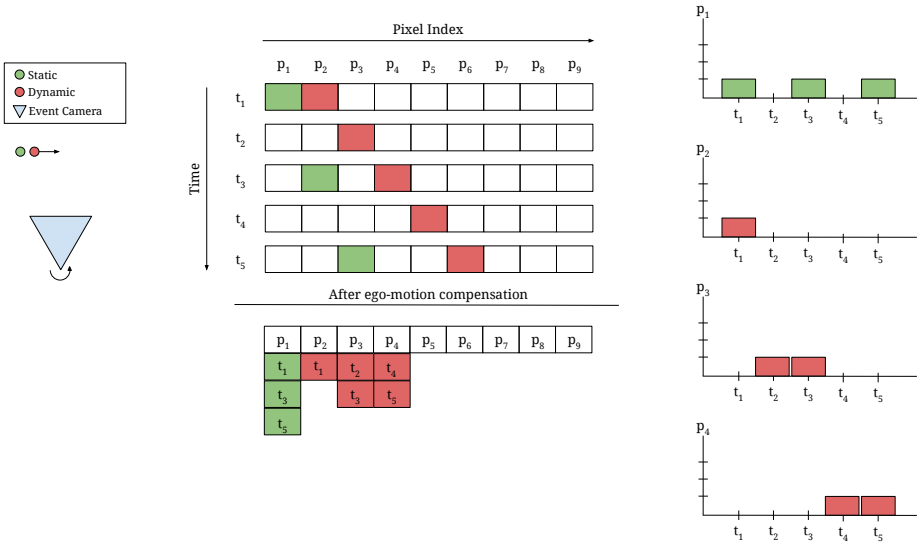
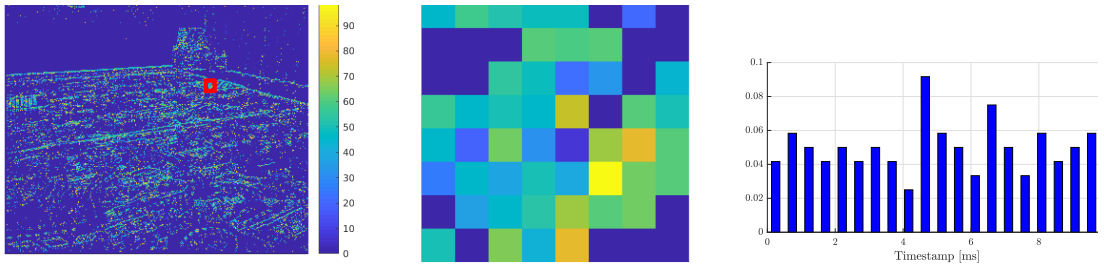
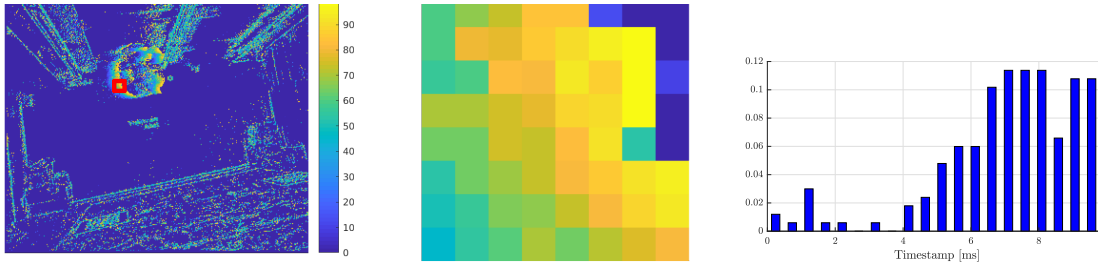


Figure D.5: A simple, yet effective example to explain the working principle of our algorithm. On the left side, a one-dimensional event camera is placed in front of two objects: a static object, represented in green, and a dynamic object, in red. The dynamic object moves from left to right, while the event camera rotates in the opposite direction. In the center of the figure, we show on top the sequence of events generated by the motion of the camera and of the dynamic object, for a fixed number of time instant (from an initial time t_1 to a final time t_5), while at the bottom the ego-motion compensated events. On the right side, finally, we report the histogram of the timestamps of all the events falling in each pixel. These histogram clearly highlight the difference in terms of temporal distribution within the time window between the events generated by the static object and the events belonging to the dynamic object.

It is important to notice that, for the sake of making this example simple enough, we only considered one type of events (either positive or negative), while in a real case each objects generated both positive and negative events, simultaneously. However, the principle can be easily extended to events with polarity. Additionally, we invite the reader to notice that not every motion can be compensated, but rather only rotations and roto-translations with respect to a planar scene. Indeed, these motions can be modeles as homographies, and the events they generate can be motion-compensated. However, since we only consider the events that fire within a very short time window, the majority of the scene moves by a very small amount of pixels, therefore we can approximate the camera motion as a homography. The mathematical description of the motion compensation algorithm is provided in Sec. D.3.1.



(a) A scene without moving objects. The patch highlighted in red in the left mean timestamp image belongs to a static part of the scene, and is reported in the center figure. On the right side, we show the histogram of all the ego-motion events belonging in such patch.



(b) A scene with one moving object. In this case, we selected a patch belonging to a dynamic part of the scene, namely a ball thrown through the field of view of the camera and moving from left to right in the frame. As one can notice, several pixels report a high mean timestamp, and the histogram of all the ego-motion compensated events belonging to the patch confirms this trend.

Figure D.6: A figure reporting the statistics of the events within a single time window for two cases: no dynamic object in the scene (top row) and one dynamic object in the scene (bottom row). For each row, we report: on the left, the mean timestamp image, with color-code shown on the right side representing the mean of the timestamps of all the events back-projected to each pixel location; in the center, a 4×4 pxl patch belonging to a static part (top row) or dynamic part (bottom row) of the scene, taken from the region highlighted in red in the mean timestamp image; on the right, the distribution of the events belonging to that patch. As one can notice, the events in a patch belonging to the static part of the scene report a fairly uniform distribution of their timestamps within the window. Conversely, the events belonging to a dynamic object are very concentrated towards one side of the window (in this case, the end).

D.2 Results

D.2.1 Evaluation of the Event-Based Obstacle Detector

In this section, we perform a quantitative evaluation of the performance and effectiveness of our algorithm to detect moving obstacles using event cameras. The first analysis we conduct is about the accuracy of the detections. We collected a large datasets of obstacle detections, including ground-truth data from an Optitrack motion-capture system, in order to characterize the detection error of our algorithm, and in Sec. D.2.1 we provide the main results for both the monocular and stereo cases.

In Sec. D.2.1, we analyze the computational cost of the algorithm, providing some details about how each component contributes to the overall time. Finally, in Sec. D.2.1 we show that our algorithm can detect different sized and shaped obstacles, while in Sec. D.2.1 we discuss the detection of multiple, simultaneous obstacles.

Accuracy

We collected a dataset of more than 250 throws, obtaining around 1200 detections, and compared the output of our event-based detector with ground-truth data from a motion-capture system. For each detection, we computed the norm of the position error and in Tab. D.1 we summarize the results.

We grouped together measurements falling within bins of size 0.5 m, with the first one starting from a distance of 0.2 m along the camera's optical axis, since the algorithm did not successfully detect the obstacles at closer ranges. In the case of the monocular detector, it was necessary to discard some of the data we collected due to the fact that, at short distances, the obstacle is often only partially visible, and therefore our monocular algorithm fails to correctly estimate its distance since it would fit a known size to a partially visible object. This issue becomes less significant as the distance to the camera increases, and after 1 m it does not significantly impact the detector's performance. On the other hand, as expected, the stereo configuration is more precise at low ranges: the further the distance between the cameras and the object, the higher the uncertainty in the triangulation and, therefore, the larger the error.

Independently of the configuration, however, the data in Tab. D.1 show that our algorithm, although not tailored towards accuracy, but rather optimized for low latency, provides measurements that are sufficiently accurate to allow a quadrotor to effectively perceive its surroundings and detect moving obstacles. Among the factors that contribute to the error in estimating the obstacles' position, the low resolution of the camera certainly plays a key role. In Sec. D.2.2 we discuss this, as well as other drawbacks of current event cameras.

	Monocular				Stereo			
Distance [m]	Mean	Median	Std. Dev.	M.A.D.	Mean	Median	Std. Dev.	M.A.D.
0.2 - 0.5 m	0.08	0.05	0.18	0.09	0.07	0.05	0.07	0.06
0.5 - 1.0 m	0.10	0.05	0.22	0.10	0.10	0.05	0.18	0.10
1.0 - 1.5 m	0.10	0.05	0.20	0.10	0.13	0.07	0.21	0.12

Table D.1: A table summarizing the accuracy of our event-based algorithm to detect moving obstacles. We analyzed both the monocular and the stereo setups, and compared the detections with ground-truth data provided by a motion-capture system. For each configuration, we report (expressed in meters) the mean, the median, the standard deviation and the maximum absolute deviation of the norm of the position error, for different ranges of distances.

Computational Cost

To quantify the computational cost of our detection algorithm, we ran an extensive evaluation by throwing objects within the field of view of the event camera, while simultaneously rotating it, and measured the time necessary to process all the events that fired within the last time window of 10 ms. Table D.2 shows the results of our evaluation, highlighting how each step of the algorithm, described in details in Sec. D.3.1, contributes to the overall computation time. Our evaluation was performed on a NVIDIA Jetson TX2 board, with the algorithm running exclusively on the CPU (i.e., the GPU available on the same board was not used at all). The numbers reported in Tab. D.2 refer to the time required to run the detection algorithm with one camera, however running multiple instances of the same algorithm for multiple cameras (as for example in the stereo case) does not affect the performance in a significant way, as the individual parts can be computed in parallel.

The most expensive part of the algorithm is given by the ego-motion compensation, which on average requires 1.31 ms (36.80% of the overall time), with a standard deviation of 0.35 ms. As one can imagine, the time necessary for this step depends on the number of events that need to be processed, and Fig. D.7 clearly shows a linear dependence between the two. To understand how many events are typically generated in real-world scenarios during our experiments, we collected some statistics about the number of events that the algorithm needs to process. The data we collected that, on average, both indoors and outdoors, the number of events belonging to a time window of 10 ms spanned between 2000 and 6000.

Another step that depends on the relative motion between the camera and the scene is the clustering of the events belonging to the dynamic obstacles. This step is necessary to understand how many objects are in the scene, and to associate each event to them. Clustering the events usually requires 0.69 ms (19.39% of the overall time), with a standard deviation of 0.20 ms. The actual processing time to cluster the events depend on the number of pixels where events belonging to dynamic obstacles fired, and Fig. D.8

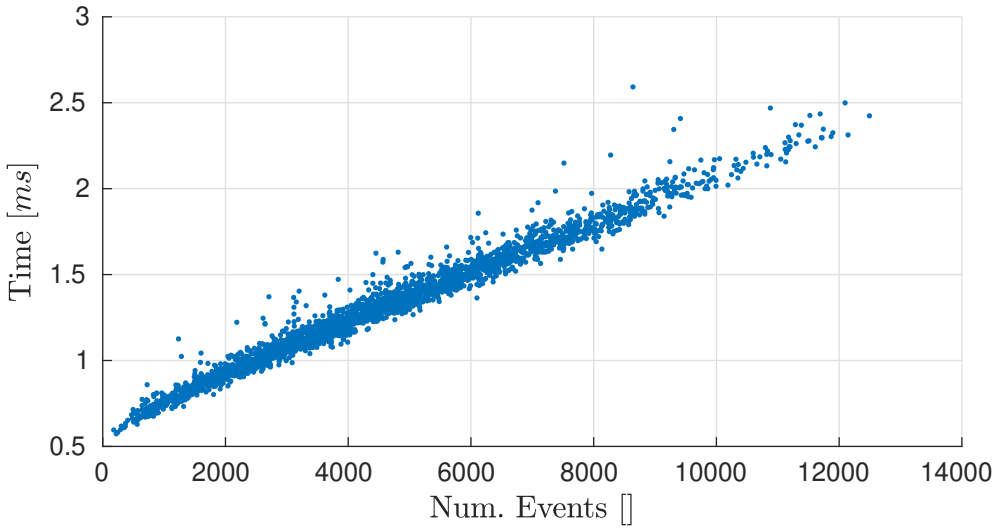


Figure D.7: Time necessary to perform the ego-motion compensation as a function of the number of events generated.

Step	μ [ms]	σ [ms]	Perc. [%]
Ego-Motion Comp.	1.31	0.35	36.80
Mean Timestamp Thresh.	0.98	0.05	27.52
Morphological Ops.	0.58	0.04	16.29
Clustering	0.69	0.20	19.39
Total	3.56	0.45	100

Table D.2: The mean μ and standard deviation σ of the computation time of the obstacle detection algorithm proposed in Sec. D.3.1.

shows how long our clustering algorithm takes as a function of the number of pixels to process.

Finally, thresholding the mean timestamp image and applying some morphological operations to the thresholded image do not depend on the amount of events to be processed (as shown by their very low standard deviations), since the entire picture has to be processed, and they require on average 0.98 ms (27.52%) and 0.58 ms (16.29% of the overall time), respectively.

It is important to notice that in our evaluation of the computational time required by the algorithm we neglected the estimation of the 3D position of the obstacle. This step requires very simple calculations (c.f. Sec. D.3.1), which are independent on the number of events generated and on average require times in order of few µs. Therefore, their impact on the overall computational time is negligible.

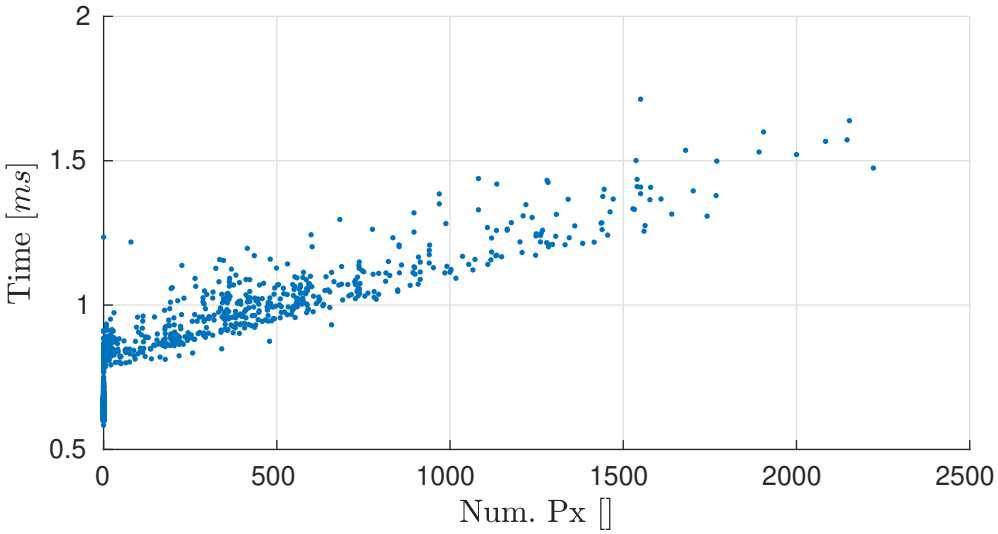


Figure D.8: Time necessary to perform the clustering of the scene's dynamic part, depending on the amount of pixels belonging to moving objects.

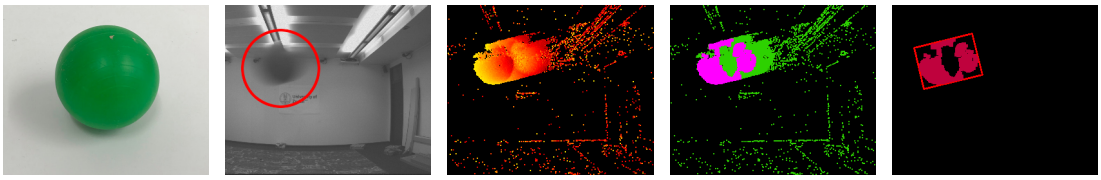
Different Types of Obstacles

The main reason for us to adopt a stereo configuration for our framework is the necessity to be able to correctly detect moving obstacles independently on their shape and size. Using a single camera, this is not possible as long as the size of the obstacle is not known in advance. Figure D.9 shows that the algorithm we propose in this paper to detect moving obstacles using two event cameras is able to detect different kinds of obstacles. In that figure, one can notice how obstacles with completely different geometries can be detected: a small ball, a box, a whiteboard marker, a frisbee, a quadrotor and a bowling pin. The first column reports a frame grabbed from the SEEM1 camera, where the object is often not clearly visible due to motion blur (we manually highlighted the region where the objects is in the frame with a red circle). The remaining columns depict the previously described steps of our detection algorithm, with the same color code used in Fig. D.4.

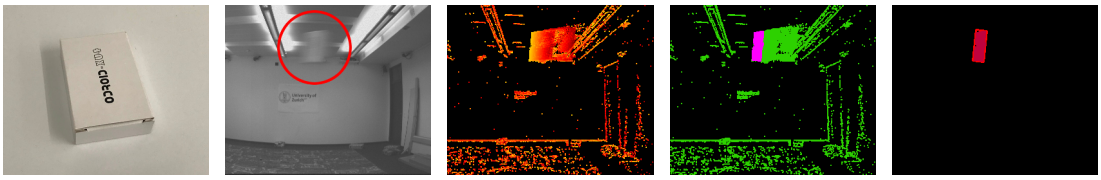
Detection of Multiple, Simultaneous Obstacles

Thanks to the clustering process proposed in Sec. D.3.1 and the measurements' association step described in Sec. D.3.1, our pipeline is able to deal with multiple obstacles moving in the scene simultaneously.

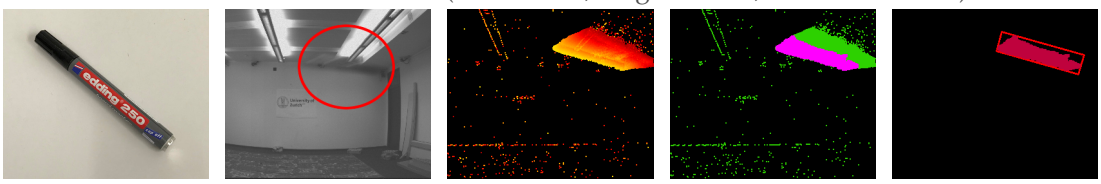
Figure D.10 shows an example where the proposed algorithm correctly detects and clusters together the events belonging to three different moving obstacles in the scene. In this case, three small-sized balls (manually highlighted by a red circle to facilitate the reader's understanding) are thrown by hand in front of the camera, and the algorithm



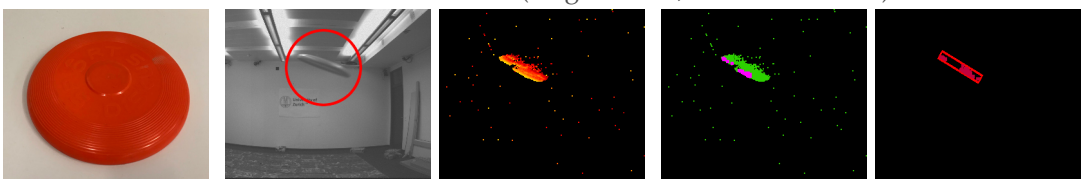
(a) A small-sized ball (radius 4 cm).



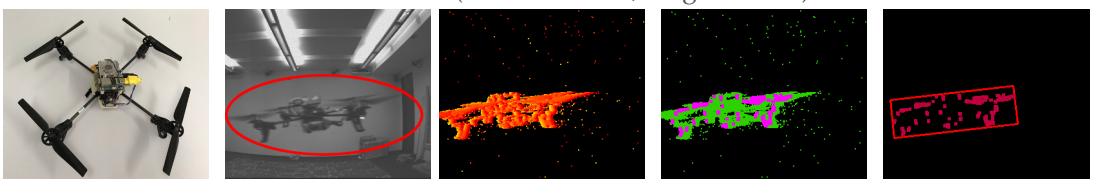
(b) A small-sized marker (width 6 cm, height 9.5 cm, thickness 2.5 cm).



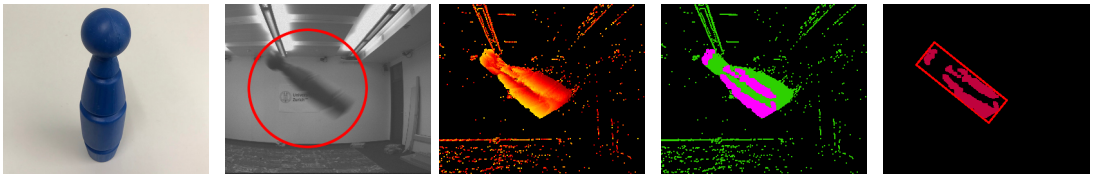
(c) A whiteboard marker (length 14 cm, thickness 1.5 cm).



(d) A frisbee (radius 13.5 cm, height 3.5 cm).

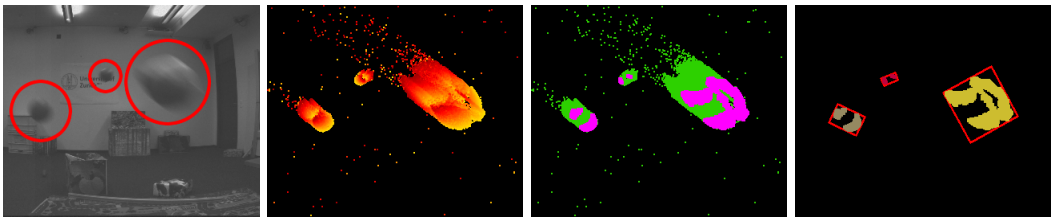


(e) A drone (tip-to-tip diagonal 60 cm, height 10 cm).



(f) A bowling pin (length 25 cm, thickness 6 cm).

Figure D.9: Our algorithm is able to detect different kind of objects, as shown in this figure. Each row of shows the detection of a different objects, depicted in the pictures in the first column. From top to bottom: a small-sized ball, a box, a whiteboard marker, a frisbee, a quadrotor and a bowling pin. These objects were detected using our stereo setup, without any prior information about their shape or size. As one can notice, the frame provided by the on-board camera (second column) presents some motion blur due to the speed of the object, which however is not a problem for our event-based detection algorithm (last column).



(a) A frame from the on-board Insightness SEEM1 camera. The three circles highlight the dynamic obstacles in the scene (three balls of different sizes).

(b) The normalized mean timestamp image generated using the events accumulated in the last time window.

(c) The normalized mean timestamp image after thresholding.

(d) Clustering of the three dynamic obstacles present in the scene.

Figure D.10: An example of our algorithm detecting and clustering multiple moving obstacles. (D.10a) The frame from the on-board camera, where three moving obstacles, manually circled in red, are visible. (D.10b) The mean-timestampe image. (D.10c) The mean-timestamp image after thresholding: green represents the static part of the scene, purple indicates events belonging to dynamic obstacles. (D.10d) Clustering of the events belonging to different dynamic obstacles.

successfully associates each event to the object they belong to.

The main limitation of our approach is due to the fact that, when two or more obstacles are very close to each other in the frame containing the events belonging to dynamic objects, it is very hard, if not impossible, to disambiguate among them. This is due to the fact that no prior information about the obstacles is used (e.g., shape or, in the stereo case, size), as well as not exploiting any intensity information (i.e., the frames from the on-board camera) in order to tell apart objects that are impossible to segment out using only events.

In our experimental evaluation this turned out not to be a real issue for the system itself, since as soon as the overlapping obstacles move away from each others, the system is able to promptly detect them and treat them as separate entities.

D.2.2 Experiments

To validate our obstacle avoidance framework, we conducted a large set of experiments in real-world scenarios. The experiments were executed in two different scenarios, one indoors, the other one outdoors. The indoor experiments were conducted within a motion-capture system, in the same setup we used in our previous work [47], and the aim was twofold: (i) collecting ground-truth data in order to verify the effectiveness of the framework in situations where a collision with the obstacle would have happened (which was checked in post-processing thanks to the data from the motion-capture);

(ii) validate the overall framework in an easier setup before moving to more complex scenarios. We used the same quadrotor platform we presented in [47] for the indoor experiments, equipped with a monocular setup. Conversely, the outdoor experiments were conducted using a different vehicle, equipped with a stereo setup, as presented in Sec. D.3.3. In the remainder of this section, we provide additional details about both the indoor (Sec. D.2.2) and outdoor (Sec. D.2.2) experiments.

Indoor Experiments

As previously mentioned, the main goal of the indoor experiments is to determine the effectiveness of our framework to avoid dynamic obstacles by determining if a collision was actually prevented by analyzing the data coming from a motion-capture system. The indoor experiments were realized using the same platform described in [47], in the monocular setup. We repeatedly threw a ball of known size towards the quadrotor, which used the event camera to detect and avoid it. Using the ground-truth measurements coming from the Optitrack motion-capture system, we could intersect the trajectory of the ball with the position where the vehicle was hovering, in order to determine if, without the execution of the escape maneuver, the ball would have hit the vehicle or not. The outcome of this analysis is that our algorithm is capable of preventing actual collisions between a flying robot and a dynamic obstacles, at relative speeds up to $10\,\mathrm{m\,s^{-1}}$, as confirmed by the ground-truth data about the objects trajectory provided by the motion-capture system.

Figure D.11 shows one of the indoor experiments, reporting four snapshot recorded with a static camera. The ball takes approximately $0.25\,\mathrm{s}$ to reach the vehicle from the moment it is thrown (Fig. D.11a). At that time, as shown in Fig. D.11d, the quadrotor already moved to the side to prevent the collision, showing that the algorithm successfully detected the ball and planned an evasive maneuver with very low latency. The experiment reported in Figure D.11, as well as other indoor experiments, are shown in the Movie S1.

Outdoor Experiments

After evaluating the performance of our framework in an indoor setup, we performed outdoor experiments using the quadrotor platform described in Sec. D.3.3, equipped with two Insightness SEEM1 cameras in a stereo setup. We executed two types of experiments, namely in a static scenario, where the vehicle hovers at a desired position, and in a dynamic scenario, where the robot flies towards a target location. In both cases, we threw different kind of objects towards the quadrotor, which only relied on the two event cameras to detect them and avoid them.

We tested the performance of our algorithm in static scenarios with different kind of

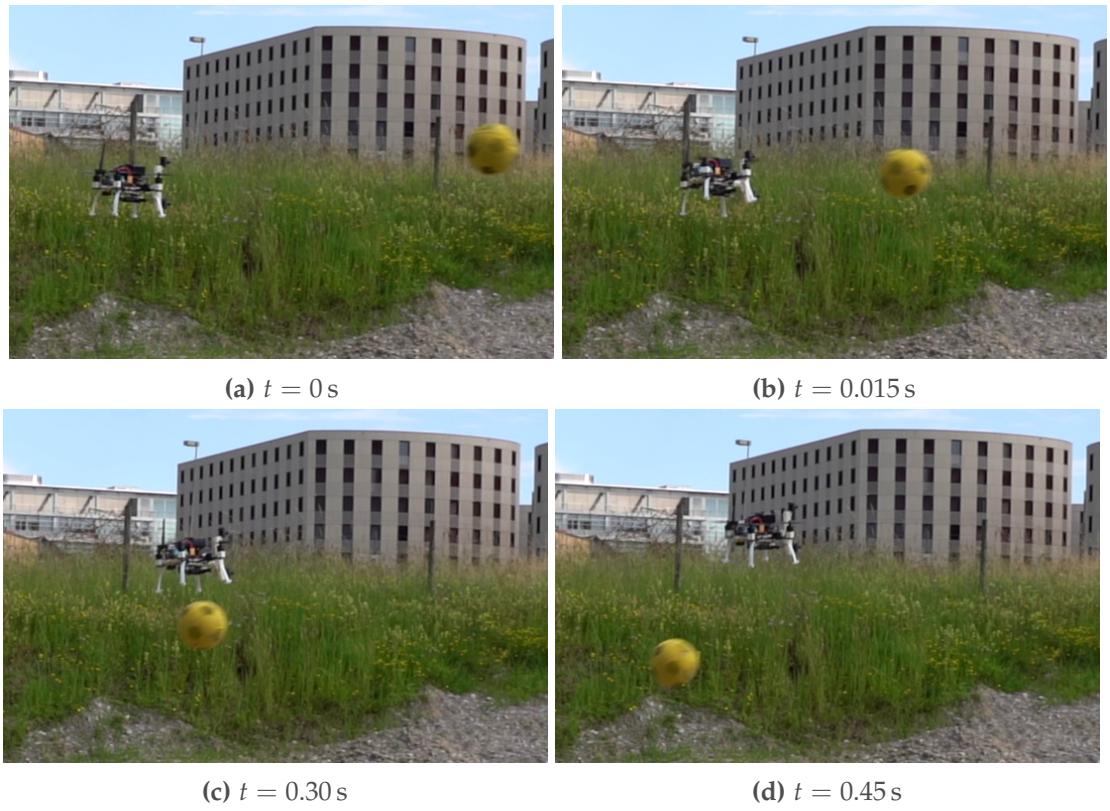
(a) $t = 0\,\mathrm{s}$

(b) $t = 0.075\,\mathrm{s}$

(c) $t = 0.15\,\mathrm{s}$

(d) $t = 0.225\,\mathrm{s}$

Figure D.11: A sequence from one of the indoor experiments. A ball is thrown towards the vehicle, equipped with a monocular event camera, which is used to detect and evade the obstacle. The ball is thrown at time $t = 0\,\mathrm{s}$, and reaches the position where the quadrotor is hovering approximately at time $t = 0.225\,\mathrm{s}$. The robot successfully detects the incoming obstacle and moves to the side to avoid it.



(a) $t = 0\,\mathrm{s}$ **(b)** $t = 0.015\,\mathrm{s}$

(c) $t = 0.30\,\mathrm{s}$ **(d)** $t = 0.45\,\mathrm{s}$

Figure D.12: A sequence from our outdoor experiments. The quadrotor is flying towards a reference goal position, when an obstacle is thrown towards it. The obstacle is successfully detected using a stereo pair of event cameras, and is avoided by moving upwards.

objects, with multiple obstacles moving towards it at the same time, as well as throwing them consecutively one after the other to benchmark the restiveness out the overall approach. The vehicle successfully manage to detect them and avoid them most of the time, although in some cases the detection was not successful and led a collision between the robot and the obstacles. In Sec. D.3.4 we discuss the major failure causes of our algorithm; nevertheless, the in outdoor experiments the algorithm successfully detected and avoid the obstacles thrown towards it more than 90% of the time. Movie S2 shows the result of our outdoor experiments in a static scenario.

Figure D.12 shows four snapshots captured from a sequence recorded in a dynamic scenario. The robot moves towards a target position, from left to right in the pictures, at a linear speed of $1.5\,\mathrm{m\,s^{-1}}$. While reaching its destination, the robot detect the yellow ball thrown towards it, and reported on the right side of Fig. D.12a. The vehicle decides to execute an evasive maneuver upwards, while keeping its travel speed towards the desired position constant. This results in a maneuver that simultaneously allows the vehicle to proceed along its task and avoid a collision. Additional experiments in a dynamic situation are shown in the Movie S3.

D.3 Materials and Methods

D.3.1 Obstacle Detection

This section describes how our event-based algorithm to detect moving obstacles works. An additional explanation of the working principle of this algorithm is provided in Movie S4.

Ego-Motion Compensation of the Events

An event camera generates events when intensity changes occur in the image. This can happen because of either moving objects or the ego-motion of the sensor. As we are only interested in avoiding moving objects, the first step is to remove all data generated by the quadrotor's ego-motion.

One way of removing ego-motion from an event stream is described by [119]. This approach does, however, utilize an optimization routine to estimate the ego-motion, which is computationally demanding and, therefore, introduces latency in the perception system. In this work we replace the optimization step with a more simple and computationally efficient ego-motion compensation algorithm. To do this, we use the IMU's angular velocity average over the time window where the events were accumulated in order to estimate the ego-rotation, and use this rotation to warp the events in the image. Our approach does not consider the translational motion of the camera, but rather assumes that the events are generated mostly by rotational motion. In order to compensate for the translational motion, it would be necessary to estimate the depth of the points generating each event, which would increase the computational complexity too much to be practical. As long as the distance to stationary objects is large enough, our system is not significantly affected by this assumption. This choice allows our pipeline to be fast enough to guarantee real-time performance, but comes at the cost of a potentially higher amount of non-compensated events. To cope with this, we tune the parameters of our algorithm, whose working principle is described below, so that it is able to filter out most of the events generated by the static part of the scene.

The first step of our algorithm requires the collection of a batch of events and IMU data over a specified time δt. Next, we average the IMU's angular velocity over δt as $\bar{\omega} = \sum_{\delta t} \omega_t$. We then apply the Rodrigues rotation algorithm to build the rotation matrix from $\bar{\omega}\delta t$ [107]. Each event e_i of the batch is then warped in the image plane by $\bar{\omega}(t_i - t_0)$, where t_0 is the time-stamp of the first event of the batch and t_i the time-stamp of event e_i. This warping is described by a field $\phi : \mathbb{R}^3 \to \mathbb{R}^3$ that warps the events' 2D displacement as $\phi(x, y, t - t_0) : (x, y, t) \to (x', t', t)$. These motion compensated events are denoted by:

$$C' = \Pi\{\phi(C)\} = \Pi\{\phi(x, y, t - t_0)\} = \{x', y', 0\} \quad \forall\{x, y, t\} \in C. \tag{D.1}$$

The original event position (x, y) is part of a discretized image plane in $\mathbb{N}^2$, while (x', y') are part of $\mathbb{R}^2$. From the warped events we construct the event-count image I, where the pixel value records the total number of events mapped to it by the event trajectory:

$$\xi_{ij} = \{\{x', y', t\} : \{x', y, , 0\} \in C', i = x', j = y'\}. \tag{D.2}$$

Here $(i, j) \in \mathbb{N}^2$ denotes the integer pixel coordinates of the discretization bin for $(x', y') \in \mathbb{R}^2$. From this we construct the event-count pixel I_{ij} as $I_{ij} = |\xi_{ij}|$, with $|A|$ being the cardinality of the set A. Next we construct the time-image T, which is also in the discretized plane $\mathbb{N}^2$. Here each pixel contains the average time-stamp of the warped events as:

$$T_{ij} = \frac{1}{I_{ij}} \sum t : t \in \xi_{ij}. \tag{D.3}$$

In order to determine which pixels belong to a moving object or the background they are each given a score $\rho(i, j) \in [-1, 1]$ for $\{i, j\} \in T$ as:

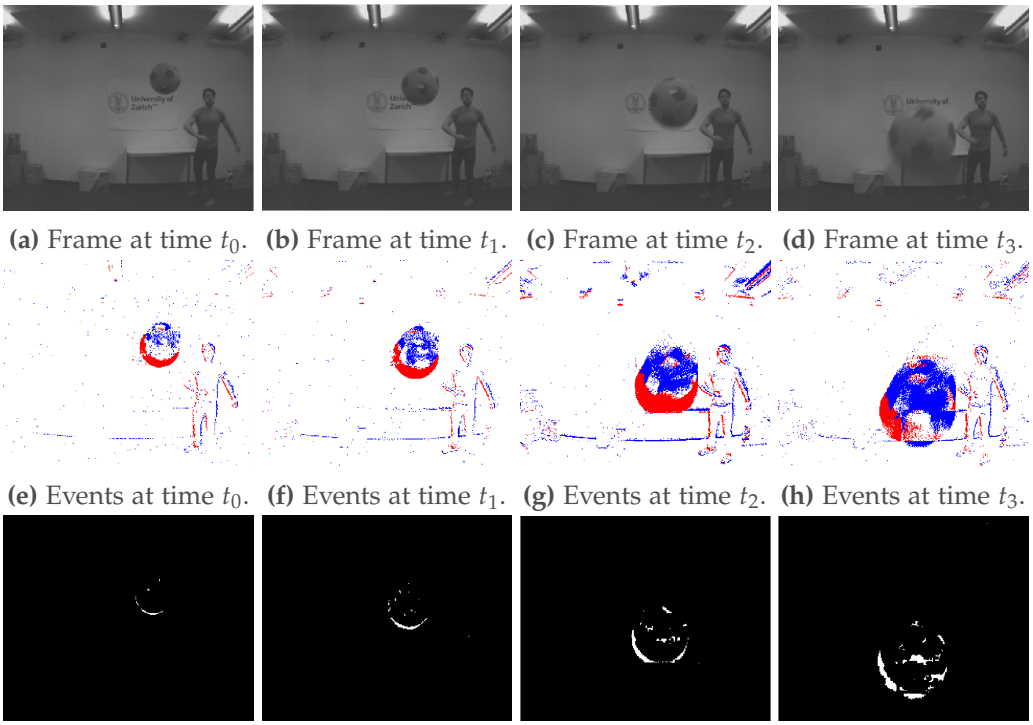
$$\rho(i, j) = \frac{T(i, j) - \text{mean}(T)}{\delta t}. \tag{D.4}$$

These scores produce the so called normalized mean time-stamp image ρ. Now, if $\rho(i, j) \geq \tau_{threshold}$, with $\tau_{threshold}$ being a specified threshold, the pixel belongs to a moving object, otherwise to the background.

While the original approach [119] uses a fixed threshold to distinguish between ego-motion generated events and those generated by a moving object, we instead use a linear function that depends on the angular velocity's magnitude, i.e. $\tau_{threshold}(\boldsymbol{\omega}) = a \cdot ||\boldsymbol{\omega}|| + b$. Here a and b are design parameters, where b regulates the threshold while the camera is static and a increases it with an increase of the angular velocity's magnitude. This has the advantage that it is easier to detect moving objects while the quadrotor is static, while still reducing the increased noise generated by faster rotational velocities. After thresholding, it can happen that some events belonging to the static part of the scene are not filtered out, generating some salt and pepper noise that we remove using morphological operations.

Figure D.4 shows our algorithm in action. All the events generated in the last time window (Fig. D.4b) are motion-compensated using the IMU and, for each pixel, we compute the normalized mean timestamp (Fig. D.4d), which is then thresholded (Fig. D.4e) to obtain a frame containing only events belonging to moving obstacles (Fig. D.4f).

The same algorithm running across different consecutive time windows is shown in



(a) Frame at time t_0. **(b)** Frame at time t_1. **(c)** Frame at time t_2. **(d)** Frame at time t_3.

(e) Events at time t_0. **(f)** Events at time t_1. **(g)** Events at time t_2. **(h)** Events at time t_3.

(i) Events belonging to moving obstacles at time t_0. **(j)** Events belonging to moving obstacles at time t_1. **(k)** Events belonging to moving obstacles at time t_2. **(l)** Events belonging to moving obstacles at time t_3.

Figure D.13: A sequence captured during one of our experiments, where the quadrotor is hovering indoors and an object is thrown towards it with the purpose of evaluating the sensing pipeline. Each column represents a different time, more specifically: $t_0 = 0\,\text{s}$ (first column), $t_1 = 0.05\,\text{s}$ (second column), $t_2 = 0.10\,\text{s}$ (third column), $t_3 = 0.15\,\text{s}$ (fourth column). The first row reports the frame captured by the on-board camera. The second row shows the events, generated by both the motion of the vehicle and the moving obstacle, collected within the last time window of size $\delta t = 10\,\text{ms}$, where blue represents positive events and red represents negative events. The third row shows the ego-motion compensated events belonging only to the dynamic part of the scene, obtained applying the algorithm described in Sec. D.3.1.

Fig. D.13. Each column corresponds to a different time, with the first row reporting the frame captured by the on-board camera, the second row the events collected in the window, and the third row the same events after the ego-motion compensation and thresholding of the normalized mean timestamp.

Obstacle Segmentation

After performing the ego-motion compensation of the events that fired in the last time window, we obtain a frame containing the location of the events belonging to the dynamic part of the scene (Fig. D.4f). It is important to note that at this point our algorithm already discarded all the static parts of the scene, with very little

computational cost. To do so with a standard camera, one has to receive at least two frames in order to be able to distinguish between static and dynamic objects, and each frame needs to be entirely processed. The output of an event camera, instead, is much more sparse, allowing us to only process the pixels where at least one event fired.

In the remainder of this section, we describe how we use a frame like the one in Fig. D.4f in order to cluster together the pixels belonging to the same object.

Clustering

The thresholded image created by the ego-motion compensation described in Section D.3.1 can include multiple moving obstacles, as well as noise. Therefore, the next step is to separate the image points of the individual objects, as well as the noise.

The goal for the system is to be capable of handling an arbitrary number of obstacles, as well as being robust towards noise. Additionally, due to the low latency requirement, the clustering has to be performed in the shortest time possible. With these requirements we evaluated different algorithms, in order to decide on the best fitting one for our system. One well known and fast clustering algorithm is the K-Means [55]. It scales well with data and has a low computational complexity of $O(kn)$, where k is the number of clusters and n the number of samples [4]. Even though the fast computation and scalability is desirable, the requirement to know the number of clusters k beforehand is a problem. Additionally, noise is included into the clusters. If noise would not be considered, one could run the K-Means for a range of k and choose the best fitting one. This would increase the computation time linearly with the increased range of k. As noise is, however, considered, the range of k would have to be large, which would limit the efficiency, or the resulting clusters accuracy is degraded drastically. These issues make this algorithm unsuitable for our approach.

Similarly to K-Means clustering, the Expectation-Maximization clustering using Gaussian Mixture Models [149] requires the knowledge of the number of clusters beforehand and, therefore, suffers some of the same drawbacks, making it unsuitable.

Next, we considered the Mean-Shift clustering algorithm [26]. Its advantage over the previously mentioned algorithms is that the number of clusters is detected automatically. The algorithm is based on finding the maxima of a density function given the discrete data samples and is an iterative approach. It can handle arbitrary feature spaces and does not depend on a predefined cluster shape. With the right kernel function it produces accurate results, but due to its time complexity of $O(n^2)$ and the iterative solution search the computation time exceeds the available time limit significantly.

Similarly to Mean-Shift, the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm detects clusters without previous knowledge about their

shape or their amount [40]. Additionally, it can handle noise by combining it into a separate category. It has an average time complexity of $O(n\log(n))$ and a maximum one of $O(n^2)$, but without the need for an iterative solution, which makes it comparatively fast. Another advantage is that its cost function can be arbitrarily chosen and, therefore, be optimized for our system. Besides the cost function, it only has two design parameters: the minimum number of data points within a cluster and the maximum cost ϵ for choosing whether a data point belongs to a given cluster. A detailed description of it is found in [40].

Optical Flow

The density of image points and their distance in the image plane depends on the objects velocity, distance to the sensor, as well as their overall size. Having only the mean time-stamp information and image position resulting from the ego-motion compensation, as described in Sec. D.3.1, makes it impossible to effectively cluster the image points of objects with different velocities and distances from the DVS. Therefore, we require additional features. One available possibility is to calculate the image points optical-flow and, therefore, get an estimate of their image-plane velocity. An added advantage is that two objects that generate image-point clusters in close proximity to each other, but move in different directions, are easier to distinguish. Ideally one would directly calculate the optical-flow from the event data, but existing algorithms for this either only produce the velocities magnitude or direction, or are extremely computationally expensive, while having a low accuracy as evaluated in [156]. Instead, we decided to use a conventional optical-flow algorithm on the unthresholded normalized mean time-stamp image produced by the ego-motion compensation. The high temporal resolution of the DVS and high update frequency of our system allows us to assume that the displacements between two frames is small and approximately constant in a region around an image point. Therefore, we use the Lucas-Kanade algorithm [7], which has the advantage that it is less sensitive to noise compared to point-wise methods and by combining the information of several nearby points it is better at handling the ambiguity of the optical-flow equations. To increase the robustness of the optical-flow we apply an averaging filter both to the input images, as well as the resulting velocity field.

Combined Clustering Algorithm

To maximize the accuracy of the clustering, we utilize all the available data information: the image position $\mathbf{p}$, the normalized mean time-stamp value ρ and the, through optical-flow estimated, velocity $\mathbf{v}$. With these quantities we constructed the DBSCAN's

cost function as:

$$C_{i,j}(\mathbf{p}, \mathbf{v}, \rho) = w_p||\mathbf{p}_i - \mathbf{p}_j|| + w_v||\mathbf{v}_i - \mathbf{v}_j|| + w_\rho|\rho_i - \rho_j|. \tag{D.5}$$

Here $\mathbf{w} = [w_p, w_v, w_\rho]^T$ is a weight vector for the influence of the individual parts.

Even though the DBSCAN algorithm is quiet efficient with a maximum data size scaling of $O(n^2)$ the computation time increases with the data size. Especially for fast moving objects, or ones which move close to the sensor, the density of the generated events and, therefore, the overall data size to be clustered increases. This leads to a far greater computation time. To combat this we perform a pre-clustering step of the image points using an eight way connected components clustering algorithm. For this we assume that two image points that are located directly next to each other in the image plane always belong to the same object. We then calculate the mean velocity of the image points belonging to the cluster, as well as the mean normalized mean time-stamp and fit a rotated rectangle around the points. The DBSCAN's cost function is adapted to the new features. Instead of using the individual point's velocity and normalized mean time-stamp, we use their corresponding mean values, while the difference in position is substituted by the minimal distance of the corresponding rectangles as:

$$C_{i,j} = w_p \, \mathrm{dist}_{\min}(r_i, r_j) + w_v||\mathbf{v}_{\mathrm{mean},i} - \mathbf{v}_{\mathrm{mean},j}|| + w_\rho|\rho_{\mathrm{mean},i} - \rho_{\mathrm{mean},j}|. \tag{D.6}$$

If two corresponding rectangles should overlap, their distance is set to zero. Instead of using rectangles, ellipses could have been used, but finding the minimal distance between two ellipses requires the root calculation of a fourth order polynomial, requiring an iterative solution, which takes drastically more time. As the connected components algorithm has a time complexity of $O(n)$ and reduces the DBSCAN's data size by orders of magnitude, the overall clustering computation time was decreased on average by a factor of 1000.

3D-Position Estimation

After receiving a set of cluster points, we first fit a rotated rectangle around them to reduce the data dimensionality. From this we get the four corner points, as well as the center position in the image-plane.

For the next step, the estimation of the obstacle's depth towards the image plane, we have to distinguish between the Monocolur and Stereo Case.

Monocular Case. As we are not able to calculate the depth of an image point from a single monocular image, we instead limit our system to objects of known size. With the added size information we can then estimate the depth of an object in the camera's

frame of reference as:

$$ {}_C\hat{z} = \frac{f\omega_{\text{real}}}{\hat{\omega}}, \tag{D.7} $$

where f is the focal length, ω_{real} the width of the object and $\hat{\omega}$ the measured side length of the fitted rectangle.

Stereo Case. For the stereo case we use the disparity between two corresponding clusters of the stereo image pair for the depth estimation. This allows the algorithm to function with objects of unknown size. To determine cluster correspondences we utilize a matching scheme minimizing the cost:

$$ C = w_p \left| x_{c,\text{top}} - x_{c,\text{bottom}} \right| + w_a \max\left(\frac{A_{\text{top}}}{A_{\text{bottom}}}, \frac{A_{\text{bottom}}}{A_{\text{top}}} \right) + w_n \max\left(\frac{n_{\text{top}}}{n_{\text{bottom}}}, \frac{n_{\text{bottom}}}{n_{\text{top}}} \right) - 2, \tag{D.8} $$

with $\mathbf{w} = (w_p, w_a, w_n)$ being weights, x_c the cluster's center's position in the image plane, A the fitted rectangle's area and n the number of cluster points. Next we use the cluster's disparity to calculate the depth as described in [60]. To increase the robustness we use the cluster's centers to estimate the depth instead of directly projecting the corner points into 3D space. Having estimated the obstacle's depth we approximate its size using the rearranged formulation as in the monocular case as:

$$ \omega_{\text{est}} = \frac{{}_C\hat{z}\hat{\omega}}{f}. \tag{D.9} $$

Image to World Projection

With the obtained obstacle's depth and size we now project the cluster's corner and center points into 3D space using:

$$ \lambda_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{K}_C\mathbf{X}_i, \tag{D.10} $$

with $\mathbf{K}$ being the intrinsic camera matrix. The points ${}_C\mathbf{X}_i$ are then transformed into the world's frame of reference by applying:

$$ \begin{bmatrix} {}_W\mathbf{X}_i \\ 1 \end{bmatrix} = \mathbf{T}_{WB}\mathbf{T}_{BC} \begin{bmatrix} {}_C\mathbf{X}_i \\ 1 \end{bmatrix}. \tag{D.11} $$

Here the center-point's depth is both increased and decreased by the obstacle's estimated size as:

$$ {}_C\hat{z}_{c,\pm} = {}_C\hat{z} \pm \omega_{\text{est}}. \tag{D.12} $$

This gives us a total of six points ${}_W\mathbf{X}_{1:6}$ representing the obstacle.

Obstacle Correspondence

In order to estimate an obstacle's velocity, we first have to determine if a newly detected obstacle corresponds to a previous one and, if this is the case, to which. This is done by finding the best match between the new obstacle's center and the predicted position of the saved obstacles' centers. This is done by finding the closest position match withing a sphere around the newly detected obstacle.

Obstacle Velocity Estimation

Once the 3D position of an obstacle has been estimated, our algorithm requires some further processing in order to provide valuable information to the planning stage, for a twofold reason: (i) the event-based detections are sometimes noisy, especially at large distances; (ii) it is necessary to estimate the obstacle's velocity, which is used to determine the avoidance direction, as well as a scaling factor for the repulsive potential field (Sec. D.3.2). To do so, we use a Kalman filter [80], with the obstacle's position estimate as the input for the measurement update. This does introduce more latency, as the Kalman filter behaves as a low-pass-filter, but the increased accuracy is in this case preferable. For this we assume a constant velocity model having as state the obstacle's position and velocity:

Since Δt is not constant, as it cannot be guaranteed that the obstacle is detected and matched in every consecutive ego-motion compensated frame, we cannot use the steady-state Kalman filter, but through its solution, obtained by solving the discrete Algebraic Riccati Equation, we can get a good initial value for $\mathbf{P}_0$.

D.3.2 Obstacle Avoidance

The primary objective of our avoidance framework is to guarantee low latency between sensing and actuation. The low latency on the perception side is guaranteed by the previously described event-based obstacle detection pipeline relying. For the overall system to be effective, however, it is necessary to reduce the latency of the decision making system responsible for driving the robot away from the detected obstacles. Based on this consideration, it is intuitive to understand that any optimization-based avoidance technique is not suited for our purpose, since numerical optimization would

introduce latency due to the non-negligible computation times. Rapid methods to compute motion primitives for aerial vehicles exist in the literature [130]. However, they present a number of drawbacks. First, it is necessary to sample both space and time to find a safe position for the robot and a suitable duration of the trajectory. Additionally, continuity in the control inputs is not always guaranteed. Finally, including this kind of methods within existing motion generation frameworks is not always trivial due to multiple reasons: it is necessary to continuously switch between the main navigation algorithm, driving the robot towards its goal, and the avoidance algorithm, steering it away from obstacles; it is not always trivial to obtain a behaviour that allows the robot to keep executing its mission (e.g., reach its goal) while simultaneously avoiding moving obstacles.

The artificial potential fields method is a natural and simple solution to all the aforementioned issues. Given a closed-form expression of the attractive and repulsive fields, it is particularly simple to compute their gradients within negligible computation time in order to generate the resulting force responsible for letting the robot move. Considering an obstacle as the source of a repulsive field also allows us to not require any sampling in space and time, since the resulting potential decides in which direction the robot should move at each moment in time. Finally, the resulting potential can be used at different levels of abstraction in order to integrate the command derived from its gradient into existing motion generation algorithms, for example as velocity or acceleration commands.

Using potential fields for path finding and obstacle avoidance has been extensively researched. This approach is, however, mostly used in 2D scenarios, whereas our system functions in 3D space. The typical approach is to build a discretized map, where each element represents the potential combined from the attractive and repulsive parts. This map building approach is feasible in 2D, but its size and the required computational power to build and analyze it increase drastically when doing so in 3D, as it increases from $O(n^2)$ to $O(n^3)$. Instead of building a map, we represent the obstacles as a struct of features, resulting in a sparse and minimal data representation. The obstacles are represented as ellipsoids, with a potential that is decaying over time. We use the estimated obstacles' position and velocity to calculate their repulsive forces at each time step. Additionally, given a reference target position, we compute the attractive force towards it. With the combined force we then produce a velocity command which is sent to the controller. Additionally, the system's behavior, when no obstacles are present, is similar to the one generated by a high-level trajectory planner driving the robot towards the desired goal location.

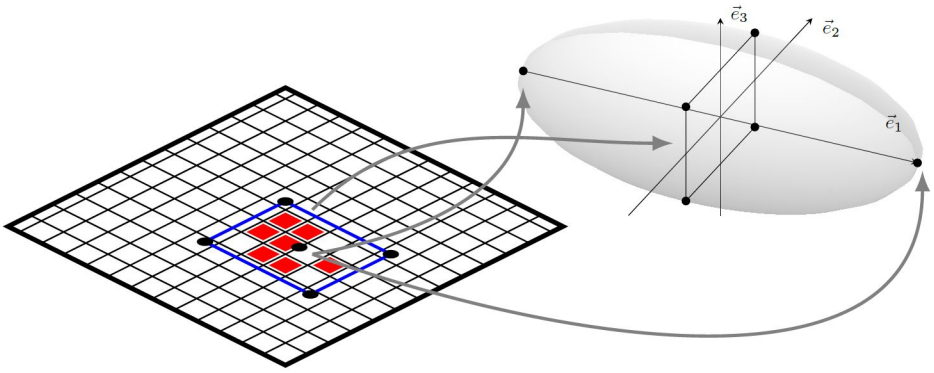


Figure D.14: Construction of the obstacle's ellipsoid in the world's frame of reference from the clustered data in the image plane. A minimal volume ellipsoid is fitted around the six projected points using an iterative approach.

Obstacles Representation

We chose to represent the obstacles as ellipsoids, as they are a good representation of the expected Gaussian error of both the position and size. Additionally, they allow us to generate a continuous repulsive force when an obstacle is detected. Using the six coordinate points ${}_W\hat{X}_{1:6}$ in Eq. D.12, we fit a minimal volume ellipsoid around them using the approach described in [123] and illustrated in Fig. D.14.

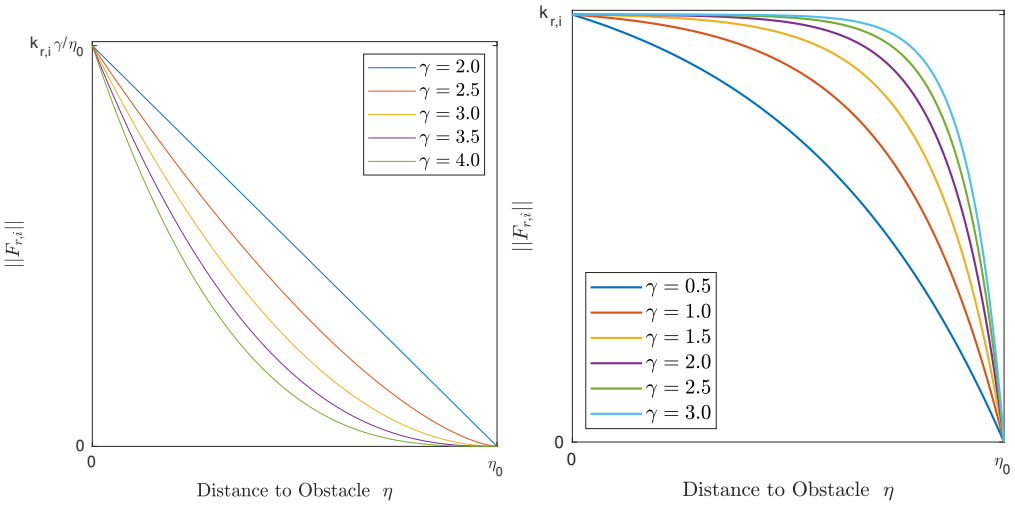
Repulsive Potential Field

Each obstacle produces a potential field $U_{r,i}$, from which we get the repulsive force $\mathbf{F}_{r,i}$ by calculating its gradient as $\mathbf{F}_{r,i} = -\nabla U_{r,i}$. One way of formulating the potential field was proposed by [86], which in turn is a modification of the original artificial potential field definition by [85], as:

$$U_{r,i}(\eta_i) = \begin{cases} k_{r,i}\left(\frac{\eta_0 - \eta_i}{\eta_0}\right)^{\gamma}, & \text{if } 0 \leq \eta_i \leq \eta_0 \\ \mathbf{0}, & \text{if } \eta_i > \eta_0 \end{cases}, \tag{D.13}$$

with a resulting force:

$$\mathbf{F}_{r,i} = -\nabla U_{r,i} = \begin{cases} \frac{k_{r,i}\gamma}{\eta_0}\left(\frac{\eta_0 - \eta_i}{\eta_0}\right)^{\gamma-1}\nabla\eta_i, & \text{if } 0 \leq \eta_i \leq \eta_0 \\ \mathbf{0}, & \text{if } \eta_i > \eta_0 \end{cases}, \tag{D.14}$$



(a) Repulsive force as described by [86]. **(b)** Our proposed repulsive force.

Figure D.15: Plots illustrating the two different types of repulsive forces described in this work.

where k_r, γ and η_0 are design parameters and η_i is the distance to the obstacle i. This kind of field does, however, produce a gradient whose magnitude increases slowly as the distance to the obstacle decreases, as shown in Figure D.15a. This has the effect that the repulsive force acting on the quadrotor only reaches significant values when the obstacle is close, or a high repulsive gain k_r has to be chosen, which might lead to an unstable, aggressive behavior.

Therefore, we propose a new formulation of the repulsive force as:

$$||\mathbf{F}_{r,i}|| = \begin{cases} k_{r,i}\left(1 - \dfrac{1 - e^{\gamma\eta_i}}{1 - e^{\gamma\eta_0}}\right), & \text{if } 0 \leq \eta_i \leq \eta_0 \\ 0, & \text{if } \eta_i > \eta_0 \end{cases}, \tag{D.15}$$

as shown in Figure D.15b. Here η_i is the minimal distance to the ellipsoid's surface of obstacle i. Through this formulation the force's magnitude is limited to a specified value k_r and increases much faster. This is desirable when evading fast moving obstacles, as compared to static ones, for which the fields described in other works were developed, as the quadrotor's dynamics require it to start evading before an obstacle comes too close, as discussed in [126].

Conventionally, the gradient of the distance towards the obstacle $\nabla\eta_i$ is responsible for the direction of the repulsive force $\mathbf{F}_{r,i}$. It points in the direction of the steepest decent of the obstacle's distance, which is the opposite direction between the quadrotor's center and the closest point on the obstacle's ellipsoid's surface. This means that an obstacle pushes the quadrotor away from it. We do, however, want to apply a different avoidance strategy. Instead, we use the obstacle's predicted velocity $\dot{\mathbf{x}}_i$ and

the distance's gradient $\nabla\eta_i$ and calculate the normalized cross product as:

$$\boldsymbol{\theta}_i = \frac{\nabla\eta_i \times \dot{\mathbf{x}}_i}{||\nabla\eta_i \times \dot{\mathbf{x}}_i||}. \tag{D.16}$$

Next, we project this vector into the plane orthogonal to the quadrotor's heading $\boldsymbol{\theta}_{\text{quadrotor}}$ as:

$$\boldsymbol{\theta}_{i,n} = \boldsymbol{\theta}_i - < \boldsymbol{\theta}_i, \boldsymbol{\theta}_{\text{quadrotor}} > \boldsymbol{\theta}_{\text{quadrotor}}. \tag{D.17}$$

With the new avoidance direction $\boldsymbol{\theta}_{i,n}$ the repulsive force $\mathbf{F}_{r,i}$ becomes:

$$\mathbf{F}_{r,i} = -\nabla U_{r,i} = \begin{cases} k_{r,i}\left(1 - \dfrac{1 - e^{\gamma\eta_i}}{1 - e^{\gamma\eta_0}}\right)\boldsymbol{\theta}_{i,n}, & \text{if } 0 \leq \eta_i \leq \eta_0 \\ \mathbf{0}, & \text{if } \eta_i > \eta_0 \end{cases}. \tag{D.18}$$

This formulation of the potential field yields to a behaviour such that, if the quadrotor is moving towards the goal location, it flies around any detected obstacle if the goal position is behind it, while if it is in hover conditions it moves in a direction orthogonal to the obstacle's velocity. Finally, we include the magnitude of the obstacle's estimated velocity $||\dot{\mathbf{x}}_i||$, into the repulsive force $\mathbf{F}_{r,i}$ as:

$$\mathbf{F}_{r,i} = -\nabla U_{r,i} = \begin{cases} ||\dot{\mathbf{x}}_i|| k_{r,i}\left(1 - \dfrac{1 - e^{\gamma\eta_i}}{1 - e^{\gamma\eta_0}}\right)\boldsymbol{\theta}_{i,n}, & \text{if } 0 \leq \eta_i \leq \eta_0 \\ \mathbf{0} & \text{if } \eta_i > \eta_0 \end{cases}. \tag{D.19}$$

By doing so, faster obstacles produce a larger repulsive force and the quadrotor will therefore perform a more aggressive avoidance maneuver. This is desirable since the faster an obstacle, the lower the avoidance time, which therefore implies the necessity for a quicker evasive maneuver.

Additionally, we ensure that the z-component of the repulsive force is always positive, namely $F_{r,i,z} = |F_{r,i,z}|$, as quadrotors with sufficiently large *thrust-to-weight* ratios are typically capable of producing larger accelerations upwards than downwards.

The repulsive constant $k_{r,i}$ is in our case dynamic and decays with time as:

$$k_{r,i}(t) = k_{r,0}\, e^{-\lambda_{decay}(t - t_{detection,i})}, \tag{D.20}$$

where $k_{r,0}$ is the initial repulsive constant, λ_{decay} a factor regulating the decay rate, t the current time and $t_{detection,i}$ the last time the specific obstacle was detected. Through this decay, obstacles are kept in case of a temporary occlusion or when they leave the camera's field of view. Their effect on the quadrotor, however, decreases as the time to their last detection increases. If $k_{r,i}$ falls below a given threshold $k_{r,\tau}$, the obstacle is removed.

Finally, the parameter η_i represents the minimal distance between the quadrotor's center to the obstacle's ellipsoid's surface minus the quadrotor's radius. The computation of the minimal distance between a point and an ellipsoid's surface is described in [37].

The total repulsive force is then the sum over all individual obstacles as:

$$\mathbf{F}_{r,total} = \sum_i \mathbf{F}_{r,i}. \tag{D.21}$$

Attractive Potential Field

The goal of the attractive potential field is to allow the vehicle to reach a desired target position and hover there until the user provides a new reference. In this work, we provide a simple formulation for the attractive potential that assumes that no static obstacles are present in the scene, i.e. the straight-line path between the robot and the obstacle is collision-free. However, one can easily replace this component of our avoidance scheme with more sophisticated methods to generate commands that drive the vehicle towards its goal. These can be based, for example, on potential field-based techniques dealing with static obstacles and local minima, which is out of the scope of this work, or completely different methods able to generate velocity or acceleration commands (as for example [24]).

For the attractive potential, we want the system to produce the same velocity towards a goal as a high-level planner would produce, if no obstacle is present, but also produce stable dynamics close to the goal. Therefore, we chose the hybrid approach of a conical and polynomial potential field [172] as:

$$U_a = \begin{cases} \frac{k_a}{(\gamma_a+1)e_0^{\gamma_a}}||\mathbf{e}||^{\gamma_a+1}, & \text{if } ||\mathbf{e}|| < e_0 \\ k_a||\mathbf{e}||, & \text{if } ||\mathbf{e}|| \geq e_0 \end{cases}. \tag{D.22}$$

This function is differentiable at e_0, i.e. the crossover distance between the two different potential fields, with $\mathbf{e}$ being the error between the goal's and quadrotor's positions, k_a the attractive constant and γ_a a design parameter. By taking its gradient we get the attractive force as:

$$\mathbf{F}_a = -\nabla U_a = \begin{cases} k_a \frac{\mathbf{e}}{||\mathbf{e}||}\left(\frac{||\mathbf{e}||}{e_0}\right)^{\gamma_a}, & \text{if } ||\mathbf{e}|| < e_0 \\ k_a \frac{\mathbf{e}}{||\mathbf{e}||}, & \text{if } ||\mathbf{e}|| \geq e_0 \end{cases}, \tag{D.23}$$

which is continuous in $\mathbf{e}$. The constant k_a regulates the output velocity $\dot{\mathbf{x}}$, see Section D.3.2, and by setting it to $k_a = ||\mathbf{v}_{des}||$ the quadrotor's velocity's magnitude is $||\dot{\mathbf{x}}|| = ||\mathbf{v}_{des}||$, while $||\mathbf{e}|| \geq e_0$ and no obstacles are present.

If we would instead solely rely on the conical potential field, the quadrotor would

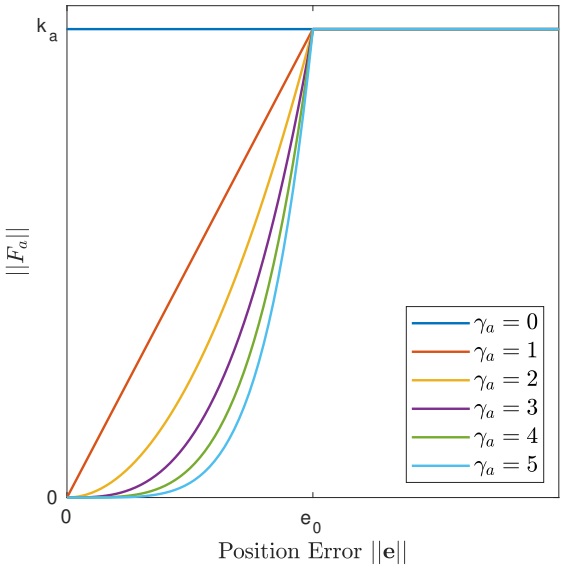


Figure D.16: Illustration of the attractive force for different values of γ_a.

start to oscillate around its goal position, as the resulting force's magnitude would be k_a, regardless of the error. The attractive force's magnitude is shown in Figure D.16. If $\gamma_a = 0$ then $||\mathbf{F}_a||$ is identical to that of the conical part, producing a constant magnitude of the attractive force, while for $\gamma_a = 1$ the magnitude goes linearly to 0. With increasing γ_a the magnitude drops faster with an increasingly large area around $||\mathbf{e}|| = 0$, where it is close to 0.

Output Velocity

The velocity is the output of our system and is given to the controller to derive the required total thrust and body-rates. From the total repulsive force $\mathbf{F}_{r,total}$ and attractive force $\mathbf{F}_a$ we get the total virtual force acting on the quadrotor as $\mathbf{F}_{total} = \mathbf{F}_{r,total} + \mathbf{F}_a$. With this force, we now have three possible design choices to calculate the quadrotor's desired velocity $\dot{\mathbf{x}}$:

$$\ddot{\mathbf{x}} = \frac{\mathbf{F}_{total}}{m} \tag{D.24}$$

$$\ddot{\mathbf{x}} = \mathbf{F}_{total} \tag{D.25}$$

$$\dot{\mathbf{x}} = \mathbf{F}_{total}, \tag{D.26}$$

where m denotes the quadrotor's mass.

Both (D.24) and (D.25) produce a first order dynamic, while (D.26) directly produces the velocity output. Introducing further dynamics into the system results in additional delays, which is undesirable since we want our system to be as responsive as possible.

We, therefore, chose (D.26) as it produces the fastest response.

D.3.3 Experimental Platform

Hardware

To validate our approach with real-world experiments, we designed a custom quadrotor platform. The main frame is a 6" Lumenier QAV-RXL, and at the end of each arm we mounted a Cobra CM2208-2000 brushless motor equipped with 6", three-bladed propeller. The vehicle is equipped with two on-board computers: (i) a Qualcomm Snapdragon Flight, used for monocular, vision-based state estimation using the provided Machine Vision SDK; (ii) a NVIDIA Jetson TX2, accompanied by an AUVIDEA J90 carrier board, running all the rest of our software stack. In this regard, the output of our framework are low-level control commands comprising the desired collective thrust and angular rates the vehicle should achieve in order to fly. These commands are sent to a Lumenier F4 AIO Flight Controller, which then produces single-rotor commands that are fed to DYS Aria 35a motor controllers.

The quadcopter is equipped with two front-facing Insighteness SEEM1 cameras, in a vertical stereo setup, connected via USB to the Jetson TX2. The SEEM1 sensor provides both frame and events, and has a QVGA resolution (320×240 pxl). In order to have a sufficiently high angular resolution, each camera has a lens providing an horizontal field of view of approximately 80°. Such small field of view is particularly low for tasks such as obstacle avoidance, where a large field of view is preferable to increase the area that the robot can sense. The choice of adopting a vertical stereo setup rather than a more common horizontal setup was driven by the necessity of maximizing the overlap between the field of view of the two cameras, while guaranteeing a sufficiently large baseline (in our case, 15 cm).

In addition to the previous sensing suite, we mounted a Teraranger EVO 60m distance sensor looking downwards. The goal of this additional sensor is to constantly monitor the height of the vehicle in order to detect whether there is any drift in the state estimate provided by the Visual-Inertial Odometry (VIO) pipeline running on the Snapdragon Flight. Whenever we detect a discrepancy beyond a manually defined threshold, the quadrotor automatically executes an emergency landing maneuver to prevent worst consequences due to the drift of the state estimate.

Software

We developed the software stack running on our quadrotor in C++ using ROS for communication among different modules. To reduce latency, we implemented the obstacle detection and avoidance algorithms within the same ROS module, so that no

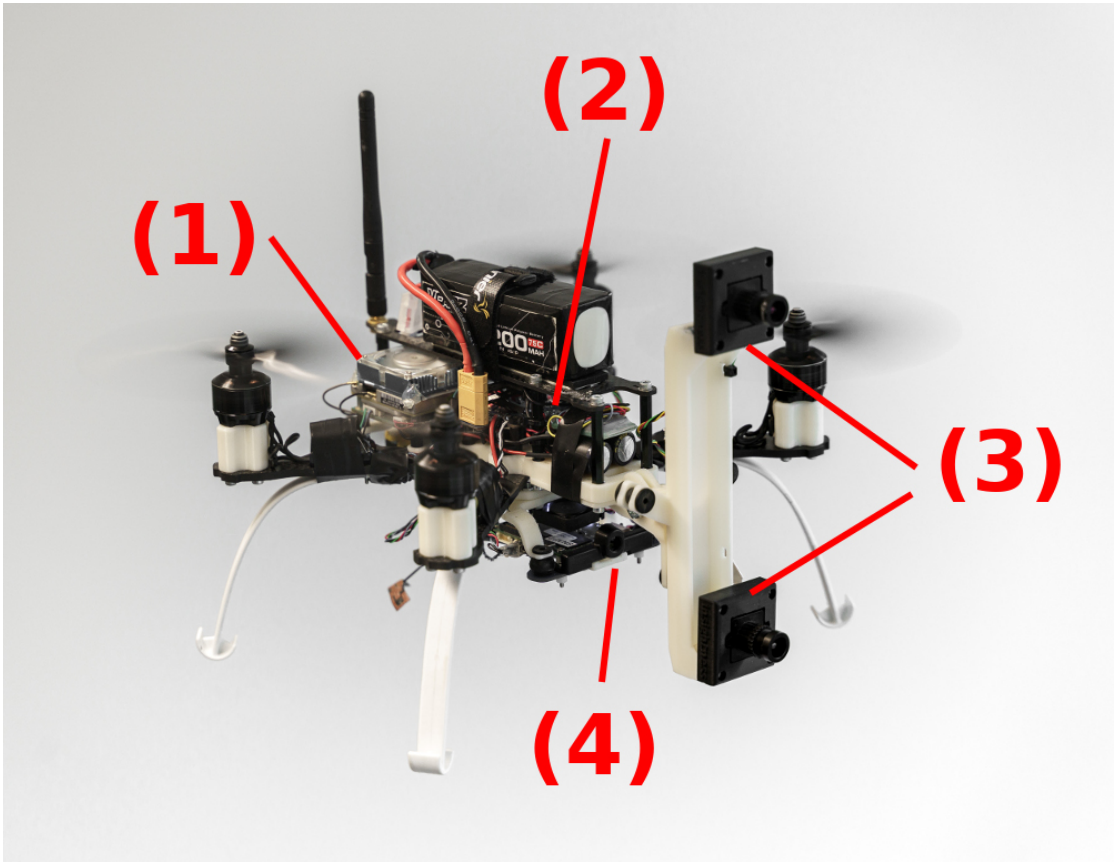


Figure D.17: The quadrotor platform we used in our outdoor experiments. The following components are highlighted in the picture: (1) the Nvidia Jetson TX2, running the obstacle detection and avoidance algorithm, as well as the high-level controller; (2) the Lumenier F4 AIO Flight Controller; (3) the two Insightness SEEM1 cameras, in a vertical stereo setup; (4) the Qualcomm Snapdragon Flight board, used for state estimation.

message exchange is necessary between the camera drivers and the code responsible for detecting moving obstacles, as well as between the latter and the planning stage. The output of this module is a velocity command, which is then fed to the position controller proposed in [44] and available as opensource [10]. The low level controller, responsible for tracking desired body rates and collective thrust, is the default one provided by the Lumenier F4 AIO Flight Controller, which then communicates with the ESCs to generate the single rotor thrusts.

In our outdoor experiments, the state of the vehicle is estimated using the Visual-Inertial Odometry pipeline provided by the Qualcomm Machine Vision SDK [11], which however only provides new estimates at camera rate (up to 30 Hz. This is not sufficient to control our vehicle with low latency, and would represent a bottleneck in the entire pipeline. In order to obtain a higher-rate state estimate, we feed the output of the VIO into an Extended Kalman Filter [103], together with IMU measurements, to obtain information about the position, orientation and velocity of the vehicle at 250 Hz.

D.3.4 Major Failure Causes, Lessons Learnt and Disadvantages of Event Cameras

As we have previously shown, event cameras allow fast, low-latency detection of moving obstacles. We discussed in Sec. D.1.2 the advantages of these novel bio-inspired neuromorphic sensors against standard camera. However, as of today, they are mostly a research-oriented sensor, and thus still require a significant engineering effort in order to solve the main issues they present.

One of the problems with current event cameras is their weight. Most of the event cameras available nowadays are larger and heavier than state-of-the-art standard cameras for robotic applications, which are typically below 50 g. The Insighteness SEEM1 is, to the best of our knowledge, the smallest event camera that also provides frames (which is particularly convenient to easily calibrate the intrinsic and extrinsic parameters of the sensor) and can be easily mounted on a quadrotor (its size is 3.5×3.5 cm and it weighs 15 g). However, its resolution (320×240 pxl, QVGA) is particularly low compared to standard cameras. This imposes the necessity to find the right trade-off between field of view and angular resolution: the larger the first, the smallest the second, which reduces the sensing range at which it is possible to reliably detect objects [47]. A small field of view, however, has a negative impact on the detection of obstacles entering the sensing range of the vehicle from the side, as for example in our outdoor dynamic experiments: the larger the field of view, the earlier the vehicle can detect and avoid obstacles moving towards it from the sides.

[10] http://rpg.ifi.uzh.ch/rpg_quadrotor_control.html
[11] https://developer.qualcomm.com/software/machine-vision-sdk

Another problem characterizing these novel sensors is their noise characteristics. Indeed, these sensors show higher noise than standard cameras, which often has a negative impact on the performance of event-based vision algorithms. In our approach, for example, in order to obtain reliable detections and to eliminate false positives caused by the sensor noise we had to significantly increase the threshold used to separate events generated by the static part of the scene from those caused by moving objects. This resulted in an obstacle detection algorithm less reactive to small relative motion, especially at large distances. For this reason, we discard all the detections reporting distances between the camera and the obstacle beyond 1.5 m.

The aforementioned reasons represent the main failure causes of our approach. In most of the cases, when our quadrotor was not able to avoid an object thrown towards it, this was due to the fact that it was detected too late, either because it entered the field of view of the camera at a distance that was too short (and therefore the vehicle could not complete the evasive maneuver in time), or because the motion of the obstacle did not generate sufficient events to allow our algorithm to detect it.

D.4 Conclusions

We presented a framework to let a quadrotor dodge fast-moving obstacles using only onboard sensing and computing. Different from state of the art, our approach relies on event cameras, novel neuromorphic sensors with reaction times of microseconds. Each pixel of an event camera reacts to changes in intensity, making this sensor a perfect fit for detecting and avoiding dynamic obstacles. Event cameras can overcome the physical limitations of standard cameras in terms of latency, but require novel algorithms to process the asynchronous stream of events they generate.

We investigated the exploitation of the temporal statistics of the event stream in order to tell apart the dynamic part of a scene, showing that it is possible to detect moving objects with a perception latency of 3.5 ms. We showed that our algorithm is capable of accurately and reliably detecting multiple simultaneous objects with different shapes and size. We combined our event-based detection algorithm with a fast strategy to generate commands that allow the vehicle to dodge incoming objects. We validated our approach with extensive experiments on a real quadrotor platform, both indoors and outdoors, demonstrating the effectiveness of the method at relative speeds up to $10\,\mathrm{m\,s^{-1}}$.

E The Foldable Drone

D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza. "The Foldable Drone: A Morphing Quadrotor that can Squeeze and Fly". In: *IEEE Robot. Autom. Lett.* 4.2 (Apr. 2019), pp. 209–216. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2885575

The Foldable Drone: A Morphing Quadrotor that can Squeeze and Fly

Davide Falanga, Kevin Kleber, Stefano Mintchev, Dario Floreano, and Davide Scaramuzza

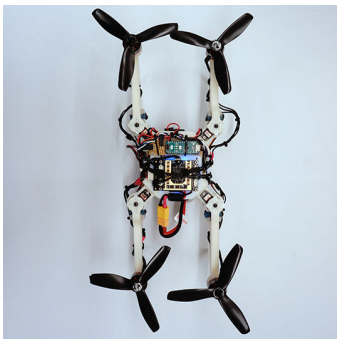
Abstract — The recent advances in state estimation, perception, and navigation algorithms have significantly contributed to the ubiquitous use of quadrotors for inspection, mapping, and aerial imaging. To further increase the versatility of quadrotors, recent works investigated the use of an adaptive morphology, which consists of modifying the shape of the vehicle during flight to suit a specific task or environment. However, these works either increase the complexity of the platform or decrease its controllability. In this paper, we propose a novel, simpler, yet effective morphing design for quadrotors consisting of a frame with four independently rotating arms that fold around the main frame. To guarantee stable flight at all times, we exploit an optimal control strategy that adapts on the fly to the drone morphology. We demonstrate the versatility of the proposed adaptive morphology in different tasks, such as negotiation of narrow gaps, close inspection of vertical surfaces, and object grasping and transportation. The experiments are performed on an actual, fully autonomous quadrotor relying solely on onboard visual-inertial sensors and compute. No external motion tracking systems and computers are used. This is the first work showing stable flight without requiring any symmetry of the morphology.

Supplementary material

All the videos of the experiments are available at:
http://youtu.be/jmKXCdEbF_E

E.1 Introduction

Quadrotors are disrupting industries ranging from agriculture to transport, security, infrastructure, entertainment, and search and rescue [52]. Their maneuverability and hovering capabilities allow them to navigate through complex structures, inspect damaged buildings, and even explore underground tunnels and caves. Yet, current quadrotors still lack the ability to adapt to different flight conditions and tasks, which is commonly observed in birds [115]. This would provide useful in complex scenarios, such as rescue and rescue missions or inspection of complex structures. For example, pigeons [143] and swifts [94] adapt their wing surface by folding in order to optimize gliding efficiency over a broad range of speeds. Pigeons have also been shown to choose different morphologies of their wings to negotiate gaps of different sizes: they fold the wings upward to negotiate relatively large vertical gaps, and fold them tight and close to their body in order to traverse narrower gaps [192]. In a similar way, a large drone could fold only when it has to fly in very cluttered environments [153]. In this way negotiation of narrow gaps can be achieved without miniaturizing the drone with consequent trade-offs in terms of flight time and payload. However, morphing quadrotors where the relative position or orientation of propellers can be modified during flight in order to extend the flight envelope remains a largely unexplored topic. The optimization of the relative orientation of the propellers [17] or the use of tiltable rotors have been investigated to increase the controllability of hovering platforms [82, 158, 157]. Although these approaches facilitate the execution of complex trajectories and manipulation tasks, they do not entail significant shape change of the frame. Quadrotors with frames that morph during flight have been investigated by Zhao et al. [195, 196], Desbiez et al. [35], Riviere et al. [153] and Zhao et al. [197] in order to negotiate narrow gaps or grasp objects, each with their own advantages and trade-offs (cf. Fig. E.2). For example, the robots in [153] and [35] can only fold into a narrow and elongated configuration (Fig. E.2a), which allows flying through narrow vertical gaps, but hampers the negotiation of tight horizontal gaps. Once folded, the quadrotor is not able to guarantee a continuous stable flight and resorts to a ballistic motion to traverse the gaps. Therefore the drone needs a significant speed at the moment it negotiates the aperture, requiring a large space before and after the gap, which might not be available in cluttered environments. Another example is the morphing aerial vehicle composed of four serially connected links equipped with propellers proposed in [195]: this robot (Fig. E.2b) is specifically conceived to wrap around objects and grasp them without the need of additional gripping device. In [196] the authors improved the morphing versatility of the drone to achieve 3D folding by departing from the standard quadrotor structure in favor of a multilink platform (Fig. E.2c). In that work a basic assumption is that each joint is actuated very slowly. The aerial transformation is time consuming, hence hampering the prompt execution of complex maneuvers. Also, the mechanical design adopted by the authors requires a large number of components (i.e., four servo motors and two rotors for each actuation unit), increasing the complexity and weight



(a) H morphology.

(b) O morphology.

(c) T morphology.

(d) Traverse of a narrow gap to enter a collapsed building.

Figure E.1: Quadrotor with morphofunctional folding capabilities. The drone can transition from the standard X configuration to task-specific morphologies: (a) H configuration to fly through narrow vertical gaps; (b) O configuration, where the drone is fully folded to fly through horizontal gaps; (c) T configuration for proximity inspection of vertical surfaces. (d) Traverse of a gap narrower than the vehicle size using the H morphology. From right to left: the quadrotor approaches the gap with the X configuration; the vehicle initiates the folding maneuver to reach the H configuration; the gap is traversed using an elongated morphology to avoid collisions.

(a) Riviere et al. [153]. **(b)** Zhao et al. [195]. **(c)** Zhao et al. [196].

Figure E.2: Examples of other morphing aerial vehicles.

of the robot. In [189] a quadrotor able to rotate and shrink its arms was presented. However, the approach proposed in that work is not able to handle non-symmetrical configurations, and only simulation results are presented. Finally, in [148] a control strategy for a flying robot with multiple degrees of freedom was proposed, and its application to a flying humanoid robot was shown.

E.1.1 Contributions

In this manuscript, we show how adaptive morphology can address the challenge of increasing quadrotors' versatility by tailoring their shape to different tasks, while limiting trade-offs such as degradation of flight time and maneuverability. The morphing approach consists of two elements working in synergy: a frame with four independently rotating arms that fold around the main frame (Fig. E.3 and E.1) and a control scheme able to take into account the current morphology of the vehicle to guarantee stable flight at all times. Each arm is connected to the main body through a servo motor and, to prevent the propellers from colliding with each other, adjacent motors have a vertical offset. This simple morphing technique allows our vehicle to preserve the structural simplicity of quadrotors without requiring complex folding mechanisms [196] or tailoring it to specific applications [195].

Differently from [153], our quadrotor is able to guarantee stable flight independently of the morphology. The key challenge to do so is the need for an adaptive control scheme able to cope in real-time with the dynamic morphology of the vehicle. Any time a new morphology is adopted, our adaptive control strategy is updated in real-time to take into account the new geometry of the robot by (i) computing the inertia matrix of the platform and (ii) solving online an Algebraic Riccati Equation (ARE) to optimize the gains of a Linear Quadratic Regulator (LQR) responsible for controlling the body rates. Also, a morphology-dependent control allocation scheme is used to compute the required propellers speeds.

We validate the effectiveness of our approach on a small-scale, autonomous, vision-based quadrotor. We show that our adaptive control strategy is able to guarantee stable in-flight morphology transition during hovering and dynamic trajectories (up

to $2\,\mathrm{m\,s^{-1}}$), without requiring any symmetry of the robot geometry. We demonstrate that the proposed morphing strategy allows a quadrotor to adapt to different tasks: (i) negotiation of narrow vertical gaps (Fig. E.1a and E.1d), (ii) negotiation of narrow horizontal gaps (Fig. E.1b), and (iii) close proximity inspection structures (Fig. E.1c). Finally, we show that the variable geometry of our quadrotor allows it to grasp and transport an object by wrapping the arms around it. Because our control and perception algorithms run directly onboard and do not need external tracking systems, we could demonstrate our drone outdoor to traverse a narrow gap and enter a partially collapsed building (see Fig. E.1d).

E.1.2 Structure of the Paper

The remainder of this paper is organized as follows. In Sec. E.2 we present our foldable quadrotor. In Sec. E.3 we introduce the adaptive control scheme used to guarantee stable flight with any morphology. In Sec. E.4 we validate our approach on a real platform and show real-world experiments. In Sec. E.5 we draw the conclusions.

E.2 Mechanical Design

Morphing systems require compromising between design complexity and shape shifting versatility. For instance, while 3D morphing frames can transition between varied and different shapes, the associated mechanical complexity could lead to cumbersome and heavy drones with limited flight time and payload [196]. 2D morphing strategies based on rotating links proved to be a reasonable compromise between feasibility and versatility [195, 153]. Avoiding singularities during morphing is another important aspect to consider in the selection of the morphing strategy to prevent complete control losses during flight [153]. We therefore decided to adopt the simple yet robust and versatile planar folding strategy composed of four folding arms as illustrated in Fig. E.3.

The mechanical design of our foldable quadrotor is composed of two main parts: (i) a central rigid body hosting the battery and the perception and control systems required for flight, and (ii) four foldable arms with rotors. Each arm has an adjustable angle $\theta_i, i = 1, ..., 4$, around the body z_b axis, which is controlled by a servomotor hosted in the central body of the drone (see Fig. E.4). The quadrotor can transition during flight form a standard X configuration (Fig. E.4, $\theta_i = \pi/4, i = 1, ..., 4$) to task-specific morphologies while trading-off flight time and maneuverability. Once the task is concluded, the quadrotor re-assumes the X configuration recovering nominal flight efficiency and maneuverability. For example, by folding the front and rear arms forward and backward respectively, the quadrotor assumes a narrow H-configuration suited to fly through narrow vertical gaps (Fig. E.1a, $\theta_1 = \theta_3 = 0$, $\theta_2 = \theta_4 = \pi/2$). However, this configuration has lower maneuverability along the roll axis than the

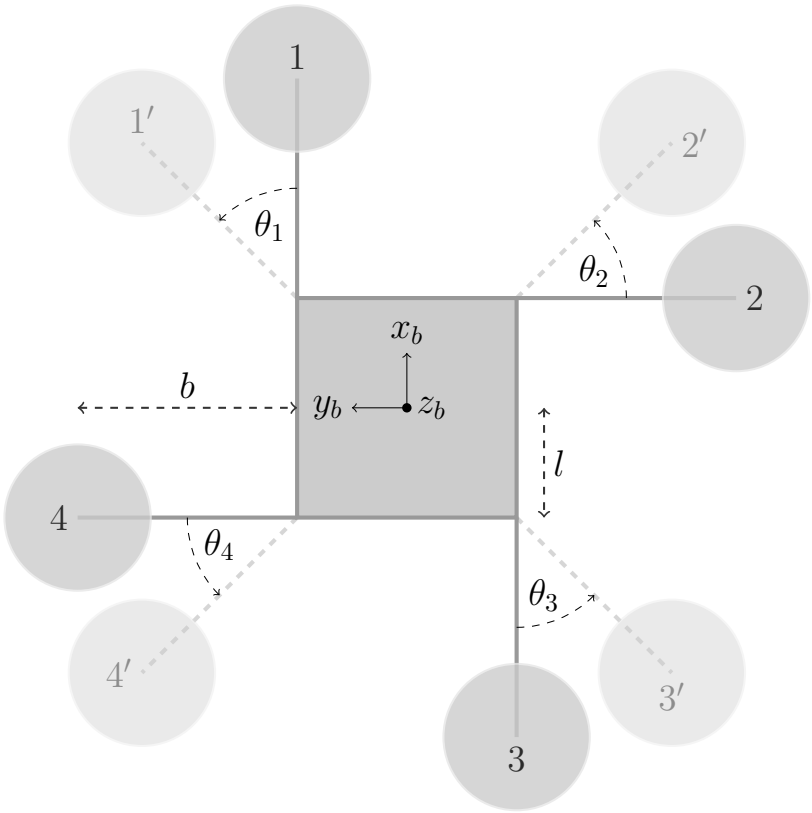


Figure E.3: Schematics of our quadrotor, able to change its morphology while flying. Each propeller is connected to the main body through an arm, which can rotate with respect to the body thanks to a servo-motor. Each arm moves independently of the others, allowing asymmetric configurations.

standard X morphology. By folding all the four arms around the central body, the quadrotor undergoes a significant size reduction along both the x and y axis (Fig. E.1b, $\theta_i = \pi, i = 1, ..., 4$). This fully folded morphology (O configuration) enables to fly through narrow horizontal gaps at the expense of major efficiency and maneuverability reductions. By folding all the arms backward, the quadrotor assumes a T configuration with the frontal part of the drone clear from propellers (Fig. E.1c, $\theta_1 = \theta_3 = \pi/2$, $\theta_2 = \theta_4 = 0$). This configuration exposes the sensorized central body of the drone, for example for the inspections of vertical surfaces.

E.3 Control

The morphology of a quadrotor has a strong impact on its mechanical properties. Specifically, the folding of the arms has a direct impact on (i) the location of the Center of Gravity (CoG) of the vehicle, (ii) the inertia tensor of the platform, and (iii) the mapping between the single rotor thrusts produced by the propellers and the forces and torques acting on the body. Therefore, a control strategy able to take into account these structural variations of the system to guarantee stable flight with any morphology is necessary.

E.3.1 Center of Gravity and Inertia

In standard quadrotors, the Center of Gravity is either considered to be located at the geometric center of the body or its offset with respect to this is estimated [93]. However, this assumption does not hold for our foldable quadrotor, as the arm angles $\theta_i, i = 1, ..., 4$, can be changed individually. The CoG, therefore, has to be recomputed when the configuration is adjusted. Similarly, the inertia matrix of the vehicle is morphology-dependent. Let $\theta_i, i = 1, ..., 4$, be the four angles of the servo motors actuating the arms. The offset $\boldsymbol{r}_{\text{CoG}} \in \mathbb{R}^3$ between the CoG and the geometric center of the vehicle is:

$$\boldsymbol{r}_{\text{CoG}} = \frac{m_{body}\boldsymbol{r}_{body} + \sum_{i=1}^{4}(m_{arm}\boldsymbol{r}_{arm,i} + m_{mot}\boldsymbol{r}_{mot,i} + m_{rot}\boldsymbol{r}_{rot,i})}{m_{body} + \sum_{i=1}^{4}(m_{arm,i} + m_{mot,i} + m_{rot,i})}, \tag{E.1}$$

where the position vectors $\boldsymbol{r}$ on the right-hand side of (E.1) are those of the corresponding part's own CoG. To simplify the computations, we refer the inertia tensor $\boldsymbol{J}$ of our foldable quadrotor with respect to the MAV's CoG. Specifically, $\boldsymbol{J}$ consists of the inertia tensors of the individual parts, which can be combined using the parallel axis theorem. We model the motors and rotors as cylinders. The arms are approximated as rectangular cuboids of length b, width w_{arm} and height h_{arm}. Finally, we model the

central body as a box having length and width l, and height h_{body}, resulting in:

$$\begin{aligned}
\boldsymbol{J}_{body} &= \frac{m_{body}}{12}\,\mathrm{diag}\left(h_{body}^2+l^2, h_{body}^2+l^2, l^2+l^2\right), \\
\boldsymbol{J}_{arm} &= \frac{m_{arm}}{12}\,\mathrm{diag}\left(w_{arm}^2+h_{arm}^2, h_{arm}^2+b^2, w_{arm}^2+b^2\right), \\
\boldsymbol{J}_{mot} &= \frac{m_{mot}}{12}\,\mathrm{diag}\left(3r_{mot}^2+h_{mot}^2, 3r_{mot}^2+h_{mot}^2, 6r_{mot}^2\right), \\
\boldsymbol{J}_{rot} &= \frac{m_{rot}}{12}\,\mathrm{diag}\left(3r_{rot}^2+h_{rot}^2, 3r_{rot}^2+h_{rot}^2, 6r_{rot}^2\right).
\end{aligned}$$

As the arms, motors, and rotors are rotated around z with respect to the body frame O_b, their inertia tensors must be rotated as well. Since the inertia tensor of a cylinder does not change when rotated around its z-axis, this rotation can be neglected for the motors' and rotors' inertia tensors. The inertia tensor of the body does not have to be rotated, as the bodies' frame of reference is fixed to O_b. Accordingly, the inertia tensors for the arms can be represented as follows:

$$\boldsymbol{J}_{arm,i} = \boldsymbol{R}_z(\theta_i)\,\boldsymbol{J}_{arm}\boldsymbol{R}_z(\theta_i)^T \qquad i \in (1,2,3,4), \tag{E.2}$$

where $\boldsymbol{R}_z$ is the rotation matrix around z depending on θ_i. With these, we derived $\boldsymbol{J}$ as:

$$\begin{aligned}
\boldsymbol{J} = {} & \boldsymbol{J}_{body} - m_{body}[\boldsymbol{r}_{body} - \boldsymbol{r}_{\mathrm{CoG}}]^2 + \\
& \sum_{i=1}^{4}(\boldsymbol{J}_{arm,i} - m_{arm}[\boldsymbol{r}_{arm,i} - \boldsymbol{r}_{\mathrm{CoG}}]^2 + \\
& \boldsymbol{J}_{mot} - m_{mot}[\boldsymbol{r}_{mot,i} - \boldsymbol{r}_{\mathrm{CoG}}]^2 + \\
& \boldsymbol{J}_{rot} - m_{rot}[\boldsymbol{r}_{rot,i} - \boldsymbol{r}_{\mathrm{CoG}}]^2),
\end{aligned} \tag{E.3}$$

with $[\boldsymbol{r}]$ being the skew-symmetric matrix of the vector $\boldsymbol{r}$.

E.3.2 Morphology-dependent Control

Once the center of gravity and the inertia matrix for the current configuration are computed, it is necessary to adapt the control scheme. The morphology-dependent controller presented in the following assumes the rotational speed of the arms around the main body to be negligible (i.e., $\dot{\theta}_i \approx 0\ \forall i$). This assumption does not represent an issue thanks to the fact that our adaptive controller continuously updates its parameters in order to cope with changes in the robot morphology. Whenever an arm is required to reach a new position, the rotation necessary to obtain it is divided into small steps and, for each step, the controller is adapted.

Since the arms can only rotate around axes parallel to the body z_b axis, the direction of the thrust produced by each propeller does not depend on the morphology. Therefore,

position control, providing the desired collective thrust t_{des}, can be achieved following the standard model derived for fixed-geometry quadrotors [106] by using state-of-the-art nonlinear controllers [44]. On the contrary, attitude control, providing the desired body torques $\varnothing_{des}$, requires a morphology-dependent and adaptive approach, since the configuration has an impact on the rotational dynamics.

The body rate controller used in this work is inspired by [45]. The dynamics of the quadrotor's body rates $\boldsymbol{\omega}$ are:

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1} \left(\boldsymbol{\tau} - \boldsymbol{\omega} \times \boldsymbol{J}\, \boldsymbol{\omega} \right). \tag{E.4}$$

We model the rotor thrusts f_i as first order systems:

$$\dot{f}_i = \frac{1}{\alpha} \left(f_{des,i} - f_i \right) \qquad i \in (1,2,3,4). \tag{E.5}$$

Assuming the coefficient relating the drag torque and the thrust of a single propeller k to be constant, for slowly changing geometry (E.5) leads to a first-order dynamics for the body torques:

$$\dot{\boldsymbol{\tau}} = \frac{1}{\alpha} \left(\boldsymbol{\tau}_{des} - \boldsymbol{\tau} \right). \tag{E.6}$$

Combining (E.4) and (E.6), we can estabilish a dynamic system with state $\boldsymbol{s} = [\boldsymbol{\omega}^T \ \boldsymbol{\tau}^T]^T$ and input $\boldsymbol{u} = \boldsymbol{\tau}_{des}$, which we linearize around $\boldsymbol{\omega} = \mathbf{0}$ and $\boldsymbol{\tau} = \mathbf{0}$ obtaining:

$$\begin{bmatrix} \dot{\boldsymbol{\omega}} \\ \dot{\boldsymbol{\tau}} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{0} & \boldsymbol{J}^{-1} \\ \mathbf{0} & -\frac{1}{\alpha} \boldsymbol{I}_3 \end{bmatrix}}_{\boldsymbol{A}} \begin{bmatrix} \boldsymbol{\omega} \\ \boldsymbol{\tau} \end{bmatrix} + \underbrace{\begin{bmatrix} \mathbf{0} \\ \frac{1}{\alpha} \boldsymbol{I}_3 \end{bmatrix}}_{\boldsymbol{B}} \boldsymbol{\tau}_{des}. \tag{E.7}$$

We designed a continuous-time infinite-horizon linear-quadratic regulator (LQR) control law $\boldsymbol{u} = \boldsymbol{u}_0 + \mathbf{K}_{\text{LQR}} \left(\boldsymbol{s} - \boldsymbol{s}_{ref} \right)$ based on (E.7) in order to minimize the cost function:

$$\mathcal{L}(\boldsymbol{s}, \boldsymbol{u}) = \int \tilde{\boldsymbol{s}}^T \boldsymbol{Q} \tilde{\boldsymbol{s}} + \tilde{\boldsymbol{u}}^T \boldsymbol{R} \tilde{\boldsymbol{u}} \, dt, \tag{E.8}$$

where $\tilde{\boldsymbol{s}} = \boldsymbol{s} - \boldsymbol{s}_{ref}$, $\tilde{\boldsymbol{u}} = \boldsymbol{u} - \boldsymbol{u}_{ref}$, and $\boldsymbol{Q}$ and $\boldsymbol{R}$ are diagonal weight matrices. Furthermore, we added two terms to the resulting control law: (i) a feedback-linearizing term $\hat{\boldsymbol{\omega}} \times \boldsymbol{J}\, \hat{\boldsymbol{\omega}}$, which compensates the coupling terms in the bodyrates dynamics (E.6); (ii) a feed-forward term $\boldsymbol{J}\, \dot{\boldsymbol{\omega}}_{des}$ to guarantee that $\boldsymbol{\omega}_{des}$ is reached with $\dot{\boldsymbol{\omega}} = \dot{\boldsymbol{\omega}}_{des}$. This results in the following control policy:

$$\boldsymbol{\tau}_{des} = \mathbf{K}_{\text{LQR}} \begin{bmatrix} \boldsymbol{\omega}_{des} - \hat{\boldsymbol{\omega}} \\ \boldsymbol{\tau}_{ref} - \hat{\boldsymbol{\tau}} \end{bmatrix} + \hat{\boldsymbol{\omega}} \times \boldsymbol{J}\, \hat{\boldsymbol{\omega}} + \boldsymbol{J}\, \dot{\boldsymbol{\omega}}_{des}, \tag{E.9}$$

where $\hat{\omega}$ and $\hat{\tau}$ are the estimates of ω and τ.

Since a stable controller is needed for changing system dynamics, we recompute the LQR gains online whenever the momentary configurations deviates significantly from the linearization point. This guarantees that the system can be stabilized in all possible configurations as long as this is feasible within the motor saturation limits. These solutions could also be precomputed and applied from a lookup-table (LUT), but our online computation has three main advantages: (i) it can adapt to the systems exact momentary state without quantization error as in a LUT; (ii) it does not require extensive re-computation on cost adjustment or other tuning; (iii) it can handle online cost changes, which might be needed to adapt to many different task scenarios.

To minimize (E.8), the following Algebraic Riccati Equation must be solved:

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{0}, \tag{E.10}$$

Leading to the optimal gain matrix $\mathbf{K}_{\mathrm{LQR}} = -\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}$. Since the arm configuration of the MAV substantially changes the inertial tensor, it has a significant influence on the body dynamics and therefore in the resulting LQR gain matrix $\mathbf{K}_{\mathrm{LQR}}$. To guarantee stable flight, the LQR gains must be adapted in real-time. This can be achieved using value iteration known from dynamic programming. Specifically, we use the approach presented in [1] for the case of a linear system resulting in an iterative algorithm to solve the discrete Algebraic Riccati Equation. The iteration process can be summarized as an iteration over the matrix $\boldsymbol{P}$ as $\boldsymbol{P}_{i+1} = \boldsymbol{A}^T\boldsymbol{P}_i + \boldsymbol{Q} - \boldsymbol{A}^T\boldsymbol{P}_i\boldsymbol{B}\left(\boldsymbol{R} - \boldsymbol{B}^T\boldsymbol{P}_i\boldsymbol{B}\right)^{-1}\boldsymbol{B}^T\boldsymbol{P}_i\boldsymbol{A}$. Termination is done upon reaching a threshold in the relative norm of the matrix $\boldsymbol{P}$ between consecutive iterations. Further details are available in [1]. To solve the problem fast enough to guarantee real-time performances, we can start from the last known value for $\boldsymbol{P}$ and therefore initialize the iterative algorithm already close to the new solution. To ensure a robust control strategy over all execute configurations, we update the dynamic model, linearization and LQR gains online based on the work in [54].

E.3.3 Control Allocation

Given the desired collective thrust t_{des} and torques $\varnothing_{des}$, it is necessary to convert those into the thrust each propeller has to produce. Since our folding scheme does not modify the direction of the thrust produced by each propeller, the collective thrust t and the torque around the body z_b axis do not depend on the configuration, and their expression follows the standard quadrotor control allocation scheme [106].

The roll and pitch torques, τ_x and τ_y respectively, can be calculated as the first two components of the cross product η between the individual rotor's distance to the CoG

and the rotor's thrust vector as:

$$\boldsymbol{\eta} = \sum_{i=1}^{4} (\boldsymbol{r}_{rotor,i} - \boldsymbol{r}_{\mathrm{CoG}}) \times f_i \, e_z. \tag{E.11}$$

This results in the following mapping between the rotor thrusts $\boldsymbol{f}$ and the roll and pitch torques:

$$\begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix} = M_{x,y} \boldsymbol{f}, \tag{E.12}$$

where $\boldsymbol{f} = \begin{bmatrix} f_1 & f_2 & f_3 & f_4 \end{bmatrix}^T$ and:

$$M_{x,y} = \begin{bmatrix} l+b\sin(\theta_1)\text{-}r_{\mathrm{CoG},y} & \text{-}l\text{-}b\cos(\theta_1)+r_{\mathrm{CoG},x} \\ \text{-}l\text{-}b\cos(\theta_2)\text{-}r_{\mathrm{CoG},y} & \text{-}l\text{-}b\sin(\theta_2)+r_{\mathrm{CoG},x} \\ \text{-}l\text{-}b\sin(\theta_3)\text{-}r_{\mathrm{CoG},y} & l+b\cos(\theta_3)+r_{\mathrm{CoG},x} \\ l+b\cos(\theta_4)\text{-}r_{\mathrm{CoG},y} & l+b\sin(\theta_4)+r_{\mathrm{CoG},x} \end{bmatrix}^T.$$

Replacing (E.12) in the control allocation matrix for a fixed-morphology quadrotor [106], we can compute the full thrust mapping equation and, by solving it with respect to $\boldsymbol{f}$, we can compute the desired single rotor thrusts.

E.4 Experiments

The supplementary video attached to this paper provides a summary of the experiments reported in the following. For an extended version of the videos reporting the experimental results we refer the reader to the project webpage:
http://rpg.ifi.uzh.ch/foldable_drone

E.4.1 Experimental Platform

Our quadrotor is made from a 3D-printed frame accommodating the electronics necessary to guarantee autonomous flight, and the servomotors to fold the arms (cf. Fig. E.4). At the end of each arm a 3 blades, 5 inch propeller is mounted on top of a Gemfan M1806L 2300KV brushless motor. The motors are controlled by a Qualcomm Snapdragon Flight Electronic Speed Controller, which receives the desired rotor speed commands from a Qualcomm Snapdragon Flight board having a quad-core 2.26 GHz ARM processor and 2GB of RAM. The Snapdragon Flight board also provides two cameras, one looking forward (used in our experiments to detect the vertical gap) and one looking down, tilted at 45° (used for state estimation and to detect the horizontal gap), and an Inertial Measurement Unit (IMU). The vehicle has a take-off weight of

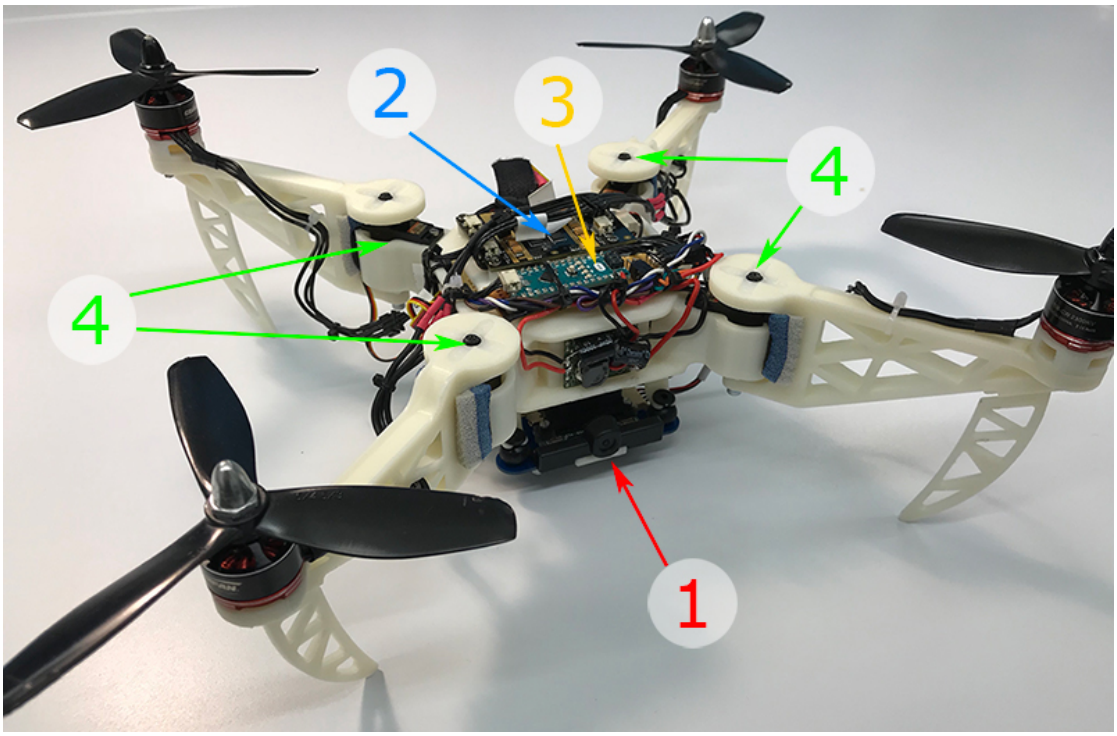


Figure E.4: A close-up picture of our foldable drone reporting the main component used. (1) The Qualcomm Snapdragon Flight onboard computer, provided with a quad-core ARM processor, 2 GB of RAM, an IMU and two cameras. (2) The Qualcomm Snapdragon Flight ESCs. (3) The Arduino Nano microcontroller. (4) The servo motors used to fold the arms.

580 g and a tip-to-tip diagonal of 47 cm.

The folding mechanism is based on the use of a servomotor directly connected to each arm. We used HiTech HS-5070MH servo motors, which provide a range of about 170°. The servomotors are commanded through an Arduino Nano micro-controller, which generates the PWM signal based on the desired angle command received by the flight controller over a USB connection. The mechanics and electronics required for morphing have an overall weight of 65g, which correspond to approximately 11% of the total weight of the platform. The combination of planar folding technique and non-backdrivable servomotors confers structural stiffness to the drone as proven by the lack of deformations and oscillations of the arms during flight. However, the current design is not crash resilient. Collisions force the arms to fold producing a torque overload on the servomotors. This limitation can be overcome with the integration of lightweight dual-stiffness mechanisms [114, 117] to decouple the arms from the servomotors during collisions.

All the computations necessary for autonomous flight are performed onboard. The state of the quadrotor (i.e., its position, orientation, linear and angular velocities) is

estimated using the Visual-Inertial Odometry pipeline provided by the Qualcomm mvSDK. Such state estimate is fed to the flight stack described in Sec. E.3, which runs onboard using ROS.

E.4.2 Morphing Trade-Offs

For each configuration presented in this work (X, T, H, O) we run in-flight experiments and performed offline evaluations in order to assess their respective advantages and trade-offs. More specifically, we are interested in:

- Flight time: the time the quadrotor can fly, which is affected by the arm configuration due to the overlap between different propellers, as well as between propellers and the main body, and due to an asymmetric usage of the motors leading to over power consumption, for example in the T configuration;
- Maximum angular acceleration as controllability index: defined as the maximum angular acceleration the robot can produce in hover around the body x_b-y_b axes;
- Size: defined as the propeller tip-to-tip distance, for both the x_b and the y_b axes.

Fig. E.5 provides a comparison among the different morphologies in terms of the aforementioned parameters, which are explained in the following. It is important to notice that the values reported in Fig. E.5 are normalized by those obtained in the X configuration. In other words, for each parameter p_i in a configuration i, Fig. E.5 reports the ratio $\frac{p_i}{p_X}$ (or its inverse, as for the size), where p_X is the same parameter evaluated in the X configuration. This is due to the fact that such a configuration is the most commonly used morphology for quadrotors, and, therefore, we took it as the reference model to evaluate advantages and disadvantages of the other configurations. Also, normalizing each value by the one obtained in the X configuration has the additional advantage of providing results that are less dependent on the specific hardware used to build our platform and allow a more fair and general comparison among different morphologies.

Flight Time

The first parameter we are interested in is the flight time each configuration is capable of providing. Since flight in dynamic conditions is highly influenced by the kind of trajectory the vehicle flies, we performed our tests in hover conditions. In this regard, we let the vehicle autonomously hover while logging the battery voltage. We performed 10 trials for each configuration using a fully charged, 3-cells, Li-Po battery. It is well known that the discharge curve for LiPo batteries is linear only within a certain region [186]; therefore, we only considered such a region to compute the flight time. As

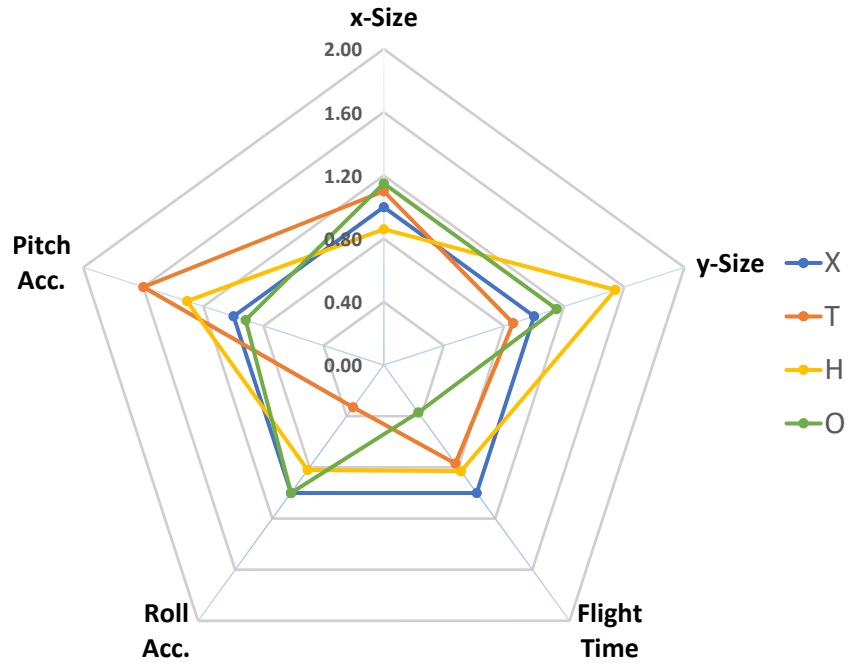


Figure E.5: Radar chart summarizing the comparison among the morphologies. We normalized each parameter to the one obtained for the X configuration, in order to provide an immediate overview about the advantages and disadvantages of each configuration compared to the classical X morphology.

expected, the X configuration is able to provide the best results and allows the vehicle to hover on average for 253 s. Changing the morphology of the drone causes a drop in the hover time of around 17%, 23%, and 63% for the H, T and O configurations, respectively. In the H configuration, this loss of endurance is partially due to the overlap between propellers. As shown in [135], when two propellers overlap, the thrust produced by the lower one depends on the vertical offset with respect to the upper one and the percentage of overlap. Our foldable quadrotor has a vertical offset between propellers of 2 cm. In the T and H configurations, the overlap is around 30% of the propeller radius, resulting in a loss of thrust for the lower propeller of around 5% [135]. The reduced flight time of the T configuration does not depend on propeller overlap, but rather on the robot geometry. In hover, rotors 1 and 2 need to rotate faster than rotors 3 and 4 due to their smaller distance to the CoG along the x_b axis (see Fig. E.3). This leads to a higher power consumption in hover with the T configuration, since in near-hover conditions the power required by each motor scales with the cube of its rotational speed [8]. Finally, in the O configuration, the flight time is reduced even more because each propeller has a 30% overlap with the main frame. Our results confirm the intuition that morphologies different from the X are less efficient, which is especially emphasized with the O configuration where the vehicle is fully folded.

Angular Acceleration

The second parameter we used to compare the different morphologies is the maximum angular acceleration the vehicle can produce around its body x_b (roll) and y_b (pitch) axes when hovering. This parameter is related to the agility and maneuverability of the platform, since it is an indicator of how fast the robot can rotate to accelerate laterally or forward. To calculate such acceleration, we first computed the maximum torque the vehicle can produce around each axis while simultaneously guaranteeing the hover thrust and satisfying the single motor thrust saturations. Then, we divided such torque by the inertia around the same rotation axis, obtaining the maximum instantaneous angular acceleration the quadrotor can produce. It is important to notice that the morphology of the robot plays a key role for this parameter and its contribution is twofold. On the one hand, folding or unfolding each arm around the main body changes the arm of the force produced by each propeller. This means that, for a propeller producing the same thrust, it can generate different torques depending on its position with respect to the fixed body. On the other hand, the inertia of the platform depends on how the arms are distributed around the main body and, the farther each propeller is with respect to the geometric center of the vehicle along one axis, the more it contributes to the inertia around the other two.

Size

Finally, we considered as last parameter of our analysis the size of the vehicle. More specifically, for each configuration we computed the tip-to-tip distance along the body's x_b and y_b. Fig. E.5 reports the results of this analysis. It is important to note that, only for the size, we considered the inverse of the ratio $\frac{p_i}{p_X}$ to guarantee consistency with the other parameters, whose normalized values larger than one indicate an improvement with respect to the X morphology.

Conclusions

The ability of switching morphology allows a quadrotor to change its shape to optimize the execution of tasks that are difficult or impossible with the X configuration, such as passing through narrow gaps, as shown in the next section. However, this comes at a cost: the standard X morphology is the most efficient and therefore should be used as long as a different morphology is not strictly required by the task at hand. Additionally, as shown by the results in Fig. E.5, reducing the size a drone by morphing does not always increase its agility. The T and H configurations, for example, are capable of providing higher angular accelerations around one of the body axes, but sacrifice their agility around the other axis. The O configuration, despite the significant size reduction, does not bring any advantage in terms of agility since the overall mass of the vehicle does not change.

E.4.3 Flight Performance

Our foldable quadrotor is able to change its morphology while flying, as shown in the attached video, where the quadrotor transitions across the four morphologies previously reported in hover conditions. Our folding scheme is able to provide stable hover flight in all such configurations, as shown in Tab. E.1, where the mean μ and standard deviation σ of the position error are shown for a flight of 60 s with each configuration. The position error does not show significant dependence on the configuration, except for the O morphology, where the overlap between the propellers and the mainframe causes a significant loss in the thrust produced by each rotor. Additionally, we performed experiments to show that, independently of the morphology, our quadrotor is able to reject external disturbances and return to the reference hover position when perturbed, as demonstrated by the results reported in the supplementary video. Finally, to show that our control scheme does not require any kind of geometric property of the morphology of the vehicle (e.g., symmetries), we performed an experiment where each servo motor is commanded to reach a randomly generated value within its range of motion.

Morphology	μ [m]			σ [m]		
	x	**y**	**z**	**x**	**y**	**z**
X	0.022	0.014	0.024	0.009	0.008	0.006
T	0.029	0.026	0.031	0.011	0.023	0.007
H	0.035	0.022	0.029	0.008	0.015	0.024
O	0.084	0.127	0.119	0.022	0.012	0.014

Table E.1: Flight performance. Statistics for the position error in hover for the four morphologies. Mean μ and standard deviation σ for the absolute value of position error for 60 s of hovering flight for each configuration, expressed in meters. Data recorded from an OptiTrack motion-capture system.

Our foldable drone is capable of changing its morphology not only in hover, but also in dynamic conditions. To this regard, the supplementary material shows experiments where the vehicle is commanded to fly a circular trajectory at a given height while changing the configuration from X to T, H, and O. We chose a circular trajectory since this requires the quadrotor to rotate both around its x_b and y_b body axes. The robot flies at a speed of $2\,\mathrm{m\,s^{-1}}$ on a circle of radius 1.5 m, at a height of 1.5 m, and is able to guarantee stable flight in all such configurations despite the large accelerations it is subject to.

Figure E.6: Traversal of an horizontal gap using the O morphology. Left: the quadrotor approaches the gap with the X configuration (time $t = 0\,\text{s}$). Center: the quadrotor starts the folding maneuver to adapt its morphology to the shape of the gap ($t = 1\,\text{s}$). Right: the gap has been traversed ($t = 2\,\text{s}$).

E.4.4 Applications

Morphing allows adaptation to a broader range of tasks and, therefore, opens the door to new applications as for example, but not limited to, flight through gaps smaller than the vehicle's silhouette, proximity inspection of surfaces and object transportation. In this section we show how our foldable quadrotor can be exploited for these tasks.

Flight Through Narrow Gaps

Previous works addressing quadrotor flight through narrow gaps have shown that an aggressive maneuver is required to align the vehicle with the gap's orientation to avoid collisions [46, 99]. Flight through arbitrarily shaped gaps using monocular vision has also been shown in [163]. In all those works, the gap has to be large enough to let the vehicle pass through. Gaps smaller than the vehicle silhouette cannot be traversed due to the fixed morphology of the robot. This increases the risk of collisions with the gap and requires a large space for the vehicle to execute and recover from this maneuver, which might not be available in unknown environments. On the contrary, an adaptive morphology enables the vehicle to pass through gaps smaller than its size by folding the arms and flying at low speed to increase safety. This also allows the robot to require a smaller free space around the gap, since no recovery maneuver is necessary. An adaptive morphology to allow a drone to pass through narrow gap was proposed in [153]. However, that vehicle can only change its morphology to one that lets it pass through vertical gaps and requires recovery maneuvering due to loss of controllability in the folded configuration.

An adaptive morphology, like the one we propose in this work, allows a quadrotor to safely fly through narrow gaps that are smaller than its size in the X configuration. For example, the H configuration lets our robot fly through vertical gaps as wide as

$2\,(l+r)$, where l is the half-size of the central body and r the propeller radius. The O configuration reduces both the width and length the robot, allowing passing through small horizontal gaps with a square shape and as wide as $2\,(l+r)$.

The results of our experiments are reported in the supplementary material for the cases of a vertical and an horizontal gap (cf. Fig E.6). In both cases, the quadrotor would collide with the frame of the gap if it would not fold the arms before traversing it. To detect the gap using on-board vision, we used the algorithm proposed in [46]. The vehicle approaches the gap with the standard X configuration, autonomously switches to a configuration that lets it traverse the gap, and finally returns to the X configuration to hover. The experiments reported in this work make use of the H configuration to traverse the vertical gap, and the O configuration to traverse the horizontal gap. The configuration to be used for each gap was decided in advance and set as a parameter in the control pipeline. The dimensions of the vertical gap are $28 \times 26\,\text{cm}$, those of the horizontal gap $32 \times 32\,\text{cm}$.

Additionally, we performed outdoor experiments to showcase the potential benefits of a quadrotor able to reduce its size in search-and-rescue missions by entering and exploring a collapsed building after traversing an aperture smaller than its size (cf. Fig. E.1d). The video of the experiments demonstrates the feasibility of our approach in a post-disaster scenario, but nevertheless this only represents a first step towards the deployment of morphing quadrotors to the field.

Close Proximity Surface Inspection

The supplementary video shows the results of an experiment highlighting the benefits of the T configuration against the X morphology for surface inspection. Indeed, the T configuration allows the robot the position the onboard front-looking camera closer than it can when the arms are placed around the main body in the X configuration (cf. Fig. E.7). If the inspection target is larger than the space between two propellers in the X morphology (i.e., the target cannot fit between two adjacent propellers), the shortest distance d from such a target that the front-looking camera can reach depends on the arm-length b and the propeller radius r, namely $d = \frac{\sqrt{2}}{2}\,(b+2r)$. Conversely, when the robot is able to fold its front arms to the side (i.e., $\theta_1 = \pi$, $\theta_2 = 0$), the camera can potentially get as close to the target as the propeller radius r. As shown in the video, the robot manages to bring the camera closer to the surface to inspect when it flies in the T configuration.

Object Grasping and Transportation

The drone can close its arms around objects to grasp and transport them. Although this strategy cannot replace a specialized end-effector, small and lightweight objects can

Figure E.7: Inspection of a surface using the T configuration (left) and the X configuration (right).

be transported without the need of additional mechanisms. The supplementary video shows our foldable drone grasping an object from the hands of a human operator by changing its morphology to the H configuration. Once the object has been grasped, the vehicle flies to a delivery point, and drops the object by simply rearranging its morphology to the X configuration.

E.5 Conclusion

In this work, we presented a simple, yet effective morphing system for quadrotors that consists of four arms that can fold around the main body. Our approach does not require symmetries in the morphology to guarantee stable flight. We showed that simple morphing mechanisms combined with adaptive control strategies are a viable solution to broaden the spectrum of applications of quadrotors. This could lead to a paradigm shift in the research community towards novel morphing aerial vehicles. However, there are still a number of unsolved research questions, such as automatic morphology selection, exploitation of the morphology for improved flight at high-speed, and novel, bio-inspired mechanical designs.

F Autonomous Landing on a Moving Platform

D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza. "Vision-based Autonomous Quadrotor Landing on a Moving Platform". In: *IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*. Oct. 2017. DOI: 10.1109/SSRR.2017.8088164

Vision-based Autonomous Quadrotor Landing on a Moving Platform

Davide Falanga, Alessio Zanchettin, Alessandro Simovic, Jeffrey Delmerico,

and Davide Scaramuzza

Abstract — We present a quadrotor system capable of autonomously landing on a moving platform using only onboard sensing and computing. We rely on state-of-the-art computer vision algorithms, multi-sensor fusion for localization of the robot, detection and motion estimation of the moving platform, and path planning for fully autonomous navigation. Our system does not require any external infrastructure, such as motion-capture systems. No prior information about the location of the moving landing target is needed. We validate our system in both synthetic and real-world experiments using low-cost and lightweight consumer hardware. To the best of our knowledge, this is the first demonstration of a fully autonomous quadrotor system capable of landing on a moving target, using only onboard sensing and computing, without relying on any external infrastructure.

Supplementary Material

Video of the experiments: https://youtu.be/Tz5ubwoAfNE

F.1 Introduction

Quadrotors are highly agile and versatile flying robots. Recent work has demonstrated their capabilities in many different applications including but not limited to: search-and-rescue, object transportation, inspection, surveillance and mapping [43, 108, 36].

Figure F.1: Our quadrotor during the landing on a moving platform.

The drawback of multirotors in general is a lower efficiency of the propulsion system when compared to other aerial vehicles, such as fixed-wing aircrafts. This limits the autonomy and utility of quadrotors as the time during which the vehicle can remain airborne is relatively short. One possible solution is to have a quadrotor autonomously land on a ground-station where its battery is charged or replaced.

Search and rescue robotics is a domain that could greatly benefit from aerial robots capable of landing autonomously on moving platforms. One day, flying robots will assist rescuers during their missions by providing an optimal platform for aerial inspection and mapping of the surroundings. Allowing these vehicles to autonomously land on predefined targets for battery charging/swapping or delivery of supplies would drastically enhance their usefulness while requiring limited or no human intervention. This would represent a major step forward in the use of autonomous robots in search-and-rescue missions, whose duration is usually significantly longer then the typical flight time of a drone.

This work focuses on the case where the ground-station moves inside a large mission area of known size (cf. Fig. F.1). Our system relies on state-of-the-art algorithms for state estimation, trajectory planning, quadrotor control and detection of the moving target, all using only onboard sensing and computing. To the best of our knowledge, this is the first demonstration of a fully autonomous quadrotor system capable of landing on a moving target, using only onboard sensing and computing, without relying on any external infrastructure.

F.1.1 Related Work

Unmanned Aerial Vehicle (UAV) landing on a desired target has been an active research field during the last decades. A large body of the literature focuses on landing a UAV on a static target, such as a predefined tag or a runway. The state of the flying vehicle is estimated using motion-capture systems [109], GPS [168, 91] or computer vision [72]. Computer vision is the most common approach when it comes to detecting the landing target [91, 168, 72, 183]. Nevertheless, solutions for detecting the target based on motion-capture systems [109], or other sensors (e.g., GPS [164]) are available in the literature. Although interesting results have been achieved, they are not necessarily applicable to dynamically moving targets in an open outdoor environment. In regard to moving targets, a number of works focused on collaboration between a flying and a ground-based vehicle to coordinate the landing maneuver [31, 132, 64]. In this work, we do not assume that the two platforms are able to communicate or coordinate a landing.

In order to detect the landing platform, most state-of-the-art works exploit computer vision from onboard cameras. Visual servoing is a valid option to some extent [92, 167]; nevertheless, it requires the landing platform to be visible throughout the entire duration of the task, the reason being that the UAV is pulled towards the goal using solely visual information from its camera. To deal with missing visual information, model-based approaches have been proposed to predict the motion of the landing target [188, 87]. Alternative solutions are realized with the use of additional sensors attached to the moving target. Among many, these sensors include Inertial Measurement Units (IMU), GPS receivers [132, 15] or infrared markers [191].

For the UAV to be truly autonomous, all of the computation necessary to achieve the goal must be performed onboard. This is by no means standard in the literature, since all the approaches mentioned before rely on external computation for state estimation, trajectory planning or control [92, 188, 87]. Additionally, GPS [164, 150, 132, 87] or motion-capture systems [92, 64] are often used for state estimation, either only while patrolling or throughout the entire task. Conversely, we rely only on onboard visual-inertial odometry for state estimation.

F.1.2 Contribution

In this paper, we present a quadrotor system capable of autonomously landing on a moving target using only onboard sensing and computing. No prior knowledge about the location of the moving landing target is needed. We exploit state-of-the-art visual-inertial odometry to estimate the state of the quadrotor itself, complemented by nonlinear control algorithms to drive the vehicle. Our system detects the landing target using an onboard camera and deals with temporarily missing visual information by

exploiting the the target's dynamical model. Therefore, no external infrastructure such as a motion-capture system is needed. We compute trajectories that take into account the dynamical model of the quadrotor and are optimal with respect to a cost function based on the energy necessary to execute it. We validate our approach in simulation as well as in real-world experiments, using low-cost, lightweight consumer hardware.

The remainder of this paper is structured as follows: Sec. F.2 provides an overview on the proposed framework and details the algorithms used to estimate the state of the quadrotor, detect and track the moving platform, plan trajectories for the aerial vehicle, and control it along these trajectories. Sec. F.3 describes the experimental platform and the simulation tools used to validate our approach, and provides the experimental results. In Sec. F.4, we discuss the proposed method and provide insights on the experiments. Finally, we draw conclusions in Sec. F.5.

F.2 System Overview

Our system makes use of the following modules:

- quadrotor state estimation (Sec. F.2.1);
- moving target detection (Sec. F.2.2);
- moving target state estimation (Sec. F.2.3);
- trajectory planning (Sec. F.2.4);
- quadrotor control (Sec. F.2.5);
- state machine (Sec. F.2.6).

Fig. F.2 provides a visual overview of these components. The modular structure of our framework allows us to easily modify or replace the algorithms inside each module without requiring changes to the others. Therefore, the one proposed in this work is a general purpose approach for landing a UAV on a moving target. It requires relatively few changes to be adapted to different platforms (e.g., fixed wings), algorithms, or scenarios.

F.2.1 Quadrotor State Estimation

We use monocular visual-inertial odometry to estimate the state of the quadrotor. More specifically, we rely on our previous work [58] for pose estimation. Pose estimates are computed at 40 Hz and fused with measurements coming from an Inertial Measurement Unit (IMU) using an Extended Kalman Filter [104] at 200 Hz. Our state estimation

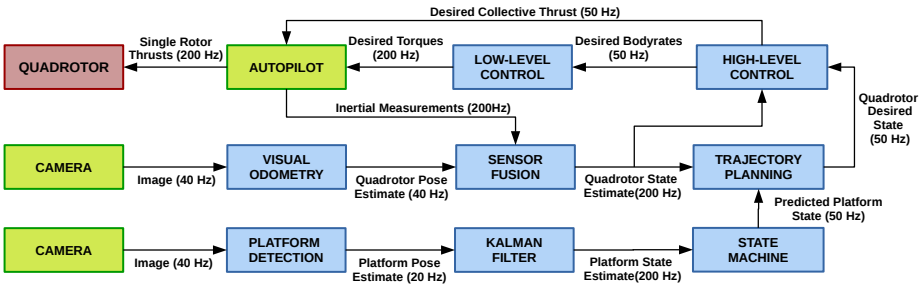


Figure F.2: A schematic representing our framework. Blue boxes represent software modules, green boxes are hardware components. The quadrotor platform is represented in red. Communication between modules happens through ROS.

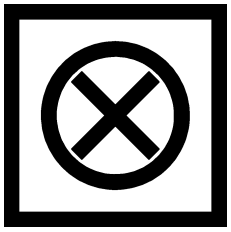


Figure F.3: The tag we used to detect the landing platform. Our framework does not strictly depend on specifics of the tag, and thanks to its modularity can easily generalize to other patterns.

pipeline provides an accurate estimate of the vehicle position, linear velocity and orientation with respect to the world frame $\{W\}$. The complete pipeline runs entirely on the onboard computer.

F.2.2 Vision-based Platform Detection

We employ onboard vision to estimate the position of the moving platform in a world frame $\{W\}$. To simplify the detection task, our moving platform is equipped with a visually distinctive tag. In this work, we leverage a tag like the one depicted in Fig. F.3. The tag consists of a black cross surrounded by a black circle with a white backdrop. Nevertheless, our framework can easily generalize to a variety of tags, as for example April Tags [139], and to different detection algorithms. Our algorithm attempts to detect the landing platform in each camera image and estimate its position in the quadrotor body frame $\{B\}$. We first convert the image from the onboard camera into a binary black-and-white image by thresholding. Next we search for the white quadrangle with the largest area. In the case where no white quadrangle is visible, the landing platform cannot be found and the detection algorithm is concluded. Conversely, if a white quadrangle is found, we search for the pattern inside the quadrangle that composes

our tag and extract its corners. More specifically, we first search for the circle and approximate it with a polygon, whose corners are used to estimate the position of the platform. If the circle is not entirely visible, we search for the four inner corners of the cross. If neither cross nor circle are visible, we use the four corners of the white quadrangle. To render our algorithm robust to outliers, we use RANSAC for geometric verification. Assuming the metric size of the tag to be known allows us to use the detected corners to solve a Perspective-n-Points (PnP) problem. In doing so, we obtain an estimate of the landing platform's position with respect to the quadrotor. Finally, we exploit the knowledge of the quadrotor's pose in world frame $\{W\}$ to transform the position of the ground platform from frame $\{B\}$ to $\{W\}$. The algorithm used to detect the platform is summarized in Alg. 1, and runs at 20 Hz on the onboard computer.

Algorithm 1 Moving landing platform detection

```
Input: Onboard camera image
Outputs: Landing platform position in {W}
binary_image ← black_and_white(camera_image)
polygons ← detect_polygons(binary_image)
landing_tag ← largest_quadrangle(polygons)
if landing_tag found then
    if circle ← detect_circle_in(landing_tag) then
        return circle.position()
    else
        if  cross ← detect_cross_in(landing_tag) then
            return cross.position()
        else
            return landing_tag.position()
else
    return 0
```

F.2.3 Platform State Estimation

The algorithm presented in Sec. F.2.2 provides an estimate of the position of the ground vehicle in the world frame W. However, the landing platform is not guaranteed to be visible at all times. To deal with missing visual detections, as well as to estimate the full state of the platform (namely the position, velocity and orientation), we use an Extended Kalman Filter [173]. We exploit a dynamic model of a ground vehicle based on *non-holonomic* movement constraints for the prediction phase [171], and consider tag detections from the onboard camera as measurements for the correction phase. For brevity reasons, we report only the main equations of the filter and refer the reader to [173] and [80] for further details.

Time Update

In the prediction step, the filter provides a prediction of the state of the moving platform based on the following non-linear equation:

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)) + \boldsymbol{w}(t), \tag{F.1}$$

where $\boldsymbol{x}(t)$, $\boldsymbol{u}(t)$ and $\boldsymbol{w}(t)$ are the state of the system, the input and the process noise, respectively. We model the process noise as white Gaussian noise, namely $\boldsymbol{w}(t) \sim \mathcal{N}(0, \sigma_w^2)$. The function $f(\boldsymbol{x}, \boldsymbol{u})$ represents the dynamical model of the moving platform:

$$\dot{p_x} = v_t \cos(\theta) \tag{F.2a}$$
$$\dot{p_y} = v_t \sin(\theta) \tag{F.2b}$$
$$\dot{p_z} = 0 \tag{F.2c}$$
$$\dot{\theta} = u_1 \tag{F.2d}$$
$$\dot{v_t} = u_2 \tag{F.2e}$$

In (F.2), p_x, p_y, p_z are the 3D coordinates of the position of the platform in the world frame {W}, θ is the angle between the x-axis of the vehicle's body frame (i.e., its forward direction) and the world x-axis, v_t is the tangential velocity of the vehicle (see Fig.F.4), and u_1 and u_2 represent the control input to the system. In our case, we assume the velocity of the platform to be constant and therefore, that the inputs u_1 and u_2 are zero all the time. If any prior information about the motion of the vehicle is available (e.g., the path along which it moves), this can be easily incorporated into the dynamical model.

Measurement Update

The correction phase is performed each time a measurement $\boldsymbol{z}_k$ (the 3D position of the moving platform) is provided by the detection algorithm, according to the following equations:

$$\hat{\boldsymbol{x}}(t) = f(\hat{\boldsymbol{x}}(t), \boldsymbol{u}(t)) + \boldsymbol{K}(t)(\boldsymbol{z}(t) - h(\hat{\boldsymbol{x}}(t)), \tag{F.3}$$

where the matrix $\boldsymbol{K}(t)$ represents the Kalman gain.

F.2.4 Trajectory Planning

We use the approach proposed in [129] to plan optimal, feasible trajectories that prevent the vehicle from colliding with obstacles. The authors of that work propose a fast

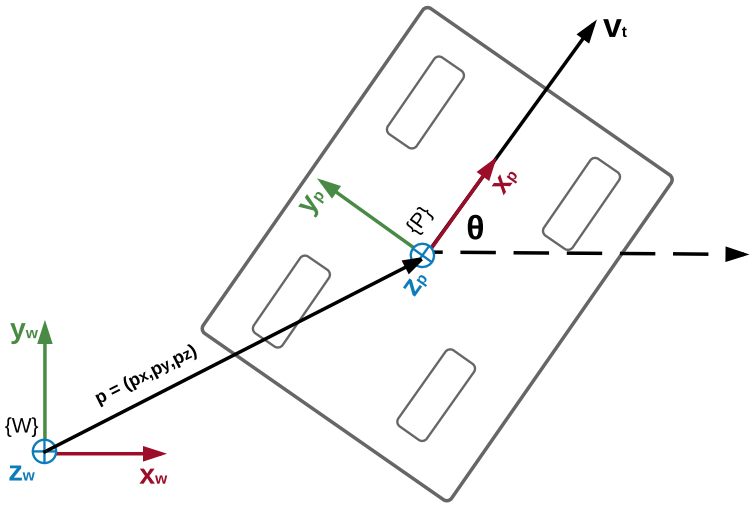


Figure F.4: A schematics representing the dynamical model of the moving platform. The world frame is indicated as {W}, while the platform body frame as {P}.

polynomial trajectory generation method that minimizes the third derivative of the position (namely, the *jerk*). Such an approach solves the minimization problem in closed form, therefore it is able to provide an optimal trajectory within a few microseconds running entirely onboard. Furthermore, the same method provides tools to verify whether the planned trajectory is feasible or not. More specifically, it allows the system to quickly check that each candidate: (i) does not exceed the physical actuation constraints of the platform, and (ii) does not collide with known obstacles (e.g., with the ground).

Additionally, during the platform following stage, we exploit the speed of the trajectory planning method [129] to provide the quadrotor with a set of feasible candidate trajectories, and we select the one with the lowest cost. Such a cost is the integral of the jerk along the trajectory, which the authors of [129] show to be an upper bound on the product of the inputs to the vehicle, namely the collective thrust and the angular velocities around the three body axes. Also, this allows us to quickly replan the desired trajectory during the platform following phase (see Sec. F.2.6). At each control cycle, we select n prediction times t_k by uniformly sampling a fixed-duration prediction horizon. For each time t_k, we predict the future state $\hat{\boldsymbol{x}}(t_k)$ that the landing platform will reach, starting from its last estimate available $\hat{\boldsymbol{x}}(t_c)$ at the current time t_c. The prediction is based on the dynamical model proposed in Sec. F.2.3. The future predicted state is used as the final state for each candidate trajectory. Out of all candidate trajectories, the one requiring a minimum amount of energy for execution is selected. We indicate the duration of the selected candidate as t_s.

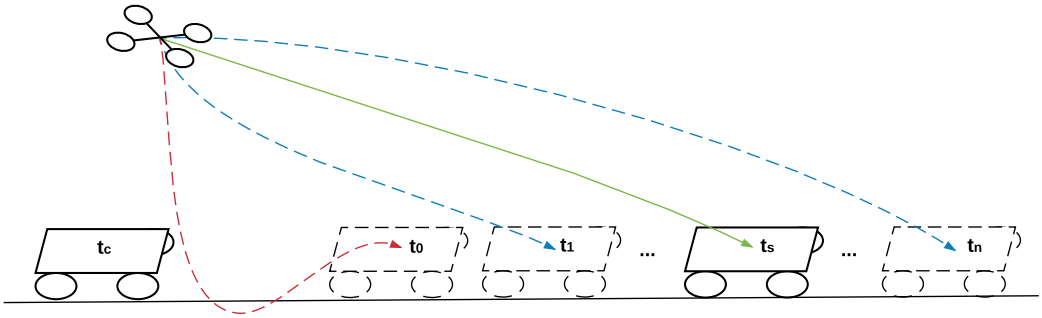


Figure F.5: An example of our planning strategy. The quadrotor plans n trajectories to reach the moving platform. Each one starts from its current position and has the ground vehicle's predicted position and velocity as final state. The future state of the moving target is predicted using its dynamical model, starting from the last estimate available from the Kalman Filter. Trajectories requiring inputs outside the allowed bounds or colliding with obstacles (e.g., with the ground), are rejected (dashed red lines in this image). We select the minimum-energy trajectory (green solid line, duration t_s) out of the set of all the feasible candidate trajectories (blue dashed lines).

F.2.5 Quadrotor Control

We use state-of-the-art, nonlinear control to drive our quadrotor along the desired trajectory. Broadly speaking, our controller is composed of a high-level controller for position and attitude corrections, and a low-level controller for reaching the required body rates. The high-level controller takes the difference between desired and estimated position, velocity, acceleration and jerk as input and returns the desired collective thrust and body rates. These body rates are passed as input to the low-level controller, which computes the necessary torques to be applied to the rigid body. The desired torques and the collective thrust are then converted to single motor thrusts. We refer the reader to our previous works [41] and [45] for further details on the dynamical model and the control algorithm used in this work.

F.2.6 State Machine

The state machine module governs the behavior of the quadrotor during the entire mission. It has four states, namely: *takeoff*, *exploration*, *platform tracking*, *landing*. Fig. F.6 depicts the state machine with its states and the respective transitions triggered by events. In the following few sections we describe each of states in more detail.

Takeoff

Our quadrotor launches from the ground and is commanded to reach a hover point within a given amount of time. During the takeoff maneuver, we rely solely on the

onboard IMU and a distance sensor. Once the vehicle is hovering, we initialize our visual odometry pipeline (Sec. F.2.1) to acquire and maintain a full state estimate. At this point, we switch the state machine to the *exploration* mode.

Exploration

The quadrotor explores an bounded area with known dimensions, flying at a given height. The vehicle autonomously computes waypoints to inspect the area and generates trajectories according to the strategy in Sec. F.2.4. This mode ends when the quadrotor detects the landing platform for the first time.

Platform Tracking

In this phase, the quadrotor follows the moving platform and attempts to reach it and fly above it. We initialize the Kalman Filter (Sec. F.2.3) after the first detection and use its output to provide the trajectory planner with waypoints. At each control cycle, the quadrotor plans a set of candidate trajectories as described in Sec. F.2.4. Once the best candidate is selected, it is compared with the previous candidate and is executed only if the final position of the two trajectories differ significantly. We consider the tracking phase concluded when the quadrotor is above the ground platform and is moving at the same velocity.

Landing

When the vehicle is close enough to the landing platform and has matched its velocity, the state machine switches to the landing mode. In this phase, we command the vehicle to start a descent at a given vertical speed, while continuing to match the speed of the landing platform along the x and y axes. We use an onboard distance sensor to estimate the relative vertical distance between the quadrotor and the ground platform. The vehicle stops the motors when the distance to the platform is below a given threshold, concluding the landing maneuver.

F.3 Experiments

F.3.1 Simulation Environment

We used RotorS [59] and Gazebo to validate our framework in simulation. We replaced the default controller provided in the simulator with our own described in Sec. F.2.5. State estimation is provided by a simulated odometry sensor and, to bring the simulation closer to real experiments, we added white Gaussian noise to the estimated state of

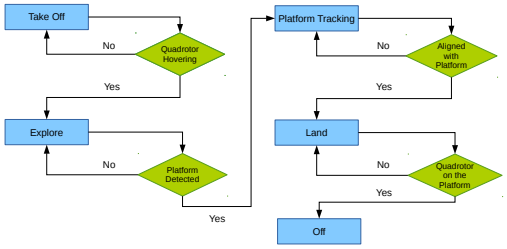


Figure F.6: The flowchart of our state machine.

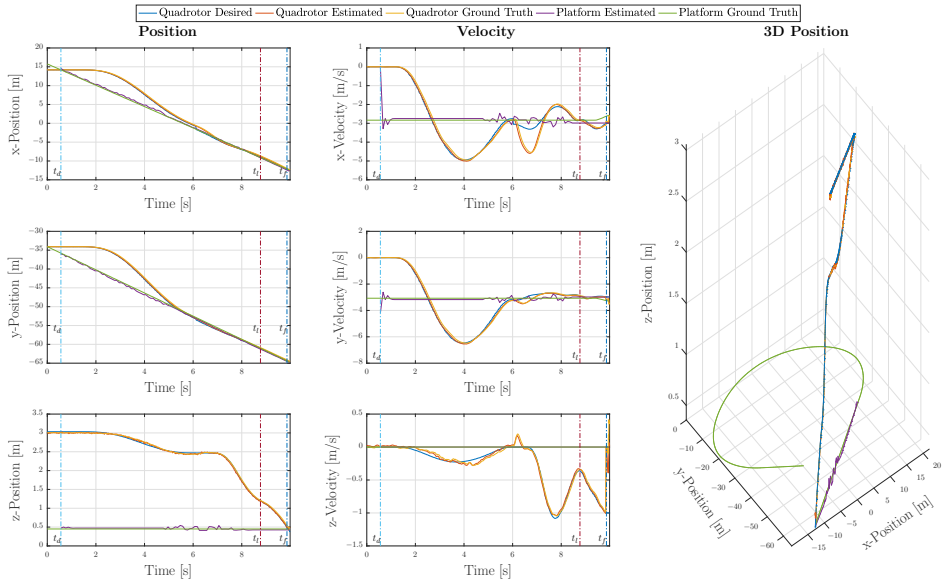


Figure F.7: The results of one of our simulations. We report data for position (left and right columns) and velocity (center column). The quadrotor starts the exploration at $t = 0$ and detects the moving platform for the first time at $t = t_d$. At this point, the tracking starts and the vehicle starts the landing phase at $t = t_l$. The maneuver is completed at $t = t_f$. The platform moves at a constant speed of $4.2\,\mathrm{m\,s^{-1}}$ along a figure-8 path.

the vehicle. Furthermore, we used an onboard simulated camera to detect the landing platform. We used a Clearpath Husky UGV simulated model as ground vehicle, on top of which we mounted the tag to be detected.

F.3.2 Simulation Results

We tested our framework using this simulation environment in a number of different scenarios. More specifically, we run simulation experiments with the landing platform moving along paths with different properties (i.e., straight line, circle, figure-8). The

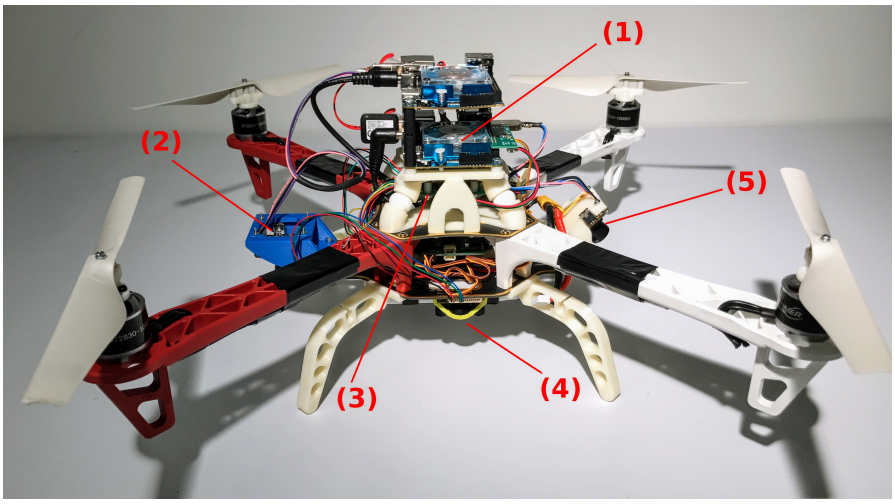


Figure F.8: The quadrotor used in our experiments. (1) The onboard computer running our algorithms. (2) The downward-looking camera used to detect the platform. (3) The PX4 autopilot. (4) The TeraRanger distance sensor. (5) The 45° angled-down camera used for visual odometry.

landing platform's speed is varied between $1\,\mathrm{m\,s^{-1}}$ and $4.2\,\mathrm{m\,s^{-1}}$. In our experiments, the quadrotor takes off from the ground and explores a pre-defined area. When the landing platform is detected, the quadrotor starts following. Once it is close enough, the quadrotor initiates the landing maneuver. The results of one of our simulated experiments are visualized in Fig. F.7.

F.3.3 Experimental Platform

For validating our framework in the real world, we used a custom-made quadrotor platform. The vehicle (cf. Fig. F.8) is constructed from both, off-the-shelf and custom 3d-printed components. We used a DJI F450 frame, equipped with RCTimer MT2830 and soft 8-inch propellers from Parrot for safety reasons. The motors are driven by *Afro Slim Electronic Speed Controllers (ESC)*. The ESCs are commanded by the *PX4 autopilot*, which also sports an Inertial Measurement Unit. Our quadrotor is equipped with two *MatrixVision mvBlueFOX-MLC200w* cameras providing an image resolution of 752×480-pixel. One camera is looking forward and is tilted down by 45°, while the second is facing towards the ground. We motivate this camera setup in Sec. F.4.2. Furthermore, we mounted a *TeraRanger One* distance sensor to estimate the scale of the vision-based pose estimation, as well as to help the quadrotor during the takeoff and landing maneuvers. The software modules of our framework (i.e., trajectory planning, quadrotor control, visual odometry and visual-inertial fusion, platform detection and tracking) run in real time in ROS on one of the two onboard *Odroid XU4* computers. The two computers are interconnected through their Ethernet ports, providing a low

latency connection. The overall weight of the platform is 1 kg, with a thrust-to-weight ratio of 1.85.

F.3.4 Landing Platform

In our real-world experiments we use a *Clearpath Jackal* [1] as ground vehicle carrying the landing platform and control it manually. In nominal conditions the platform can reach a maximum speed of $2\,\mathrm{m\,s^{-1}}$. We installed a $150 \times 150\,\mathrm{cm}$ wooden landing pad equipped with the tag on the top of the vehicle, reducing its maximum speed to approximately $1.5\,\mathrm{m\,s^{-1}}$ due to the additional weight.

F.3.5 Real Experiments Results

We demonstrated our framework in a number of real experiments using the previously describe quadrotor platform. Similarly to our simulations, we tested the effectiveness of the proposed approach in different scenarios. More specifically, we had the landing platform moving along different paths, at different speeds. Fig. F.9 reports the results for one of the experiments we conducted, with the landing platform moving on a straight line at $1.2\,\mathrm{m\,s^{-1}}$. The choice of such a speed is not due to limitations of our quadrotor system, but rather to the maneuverability of the ground robot used as moving target. The quadrotor starts the exploration at $t = 0$. The first platform detection happens at $t = t_d$, when the quadrotor starts the following phase. At $t = t_l$, the state machine detects that the vehicle is above the platform and moves at approximately its speed, entering the landing stage. Finally, the quadrotor reaches the platform at $t = t_f$ and the maneuver is completed. For brevity reasons, we do not report any comparison between the estimated state of the quadrotor and ground-truth. We refer the reader to [58] for an extensive evaluation of the performance of our visual odometry pipeline.

F.4 Discussions

F.4.1 Generality of the Framework

With the modular architecture of our framework as presented in Sec. F.2 it is straightforward to adapt it for different scenarios: Depending on the severity, changes in the hardware setup might require adjustments of the state estimation (Sec. F.2.1 and quadrotor control (Sec. F.2.5) modules. In most cases, however, a re-tuning of the low- and high-level controller's parameters should suffice. Should it be required to equip the landing platform with a different kind of tag or markers or even active beacons, all necessary changes are confined to the target detection module (Sec. F.2.2). Likewise,

[1] https://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/

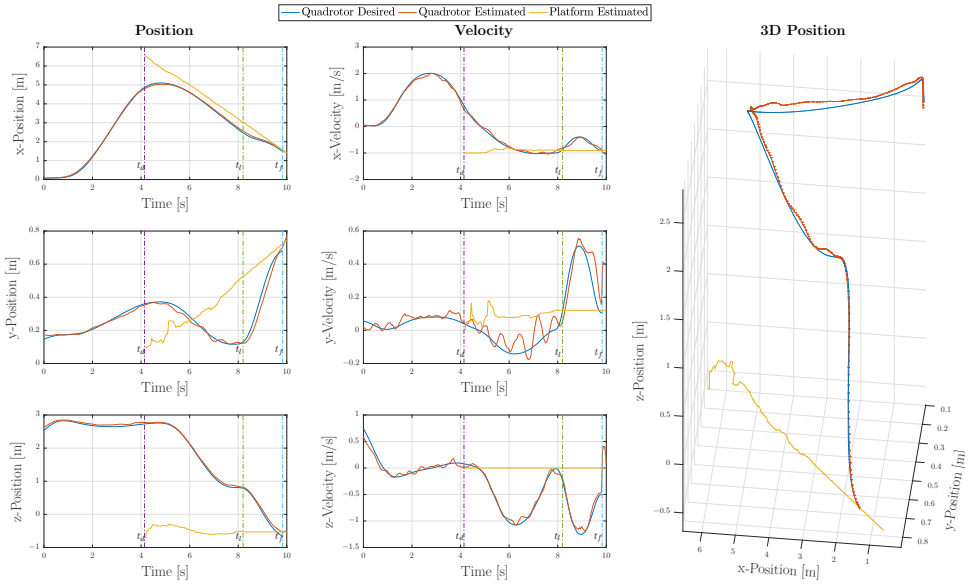


Figure F.9: The results of one of real experiments. We report data for position (left and right columns) and velocity (center column). The quadrotor starts the exploration at $t = 0$ and detects the moving platform for the first time at $t = t_d$, starting the tracking stage begins. During the landing stage, starting at $t = t_l$, the platform exits the field of view of the camera and the prediction of its motion is based solely on the dynamical model. The maneuver is completed at $t = t_f$. The platform moves at a constant speed of $1.2\,\mathrm{m\,s^{-1}}$ along a straight line.

the module for estimating the moving target's state (Sec. F.2.3) can be modified in cases where the landing platform exhibits drastically different dynamics than our model described in Eq (F.2). By modifying the state machine (Sec. F.2.6), the nature of the task can be altered. An example of such are autonomous reconnaissance missions where the quadrotor takes off and lands on a larger mobile robot. Another use case might be to have the quadrotor, or any kind of UAV for that matter, track a ground vehicle and provide a bird's view of it.

F.4.2 Motivation of the Vision Hardware Setup

Our experimental platform is equipped with two cameras, one forward-facing and is tilted down by 45° for visual odometry, one downward-looking to detect the platform. We chose this setup in order to have robust state estimation and to better detect the platform. Indeed, when the quadrotor is close to the ground vehicle, the image from the camera looking downwards contains mainly, if not only, the moving platform. Thus our visual odometry pipeline would estimate only the relative motion with respect to the platform instead of a static world frame. A forward-looking camera solves this problem.

Table F.1: Computation time statistics for our onboard, vision-based platform detection algorithm.

	Mean	Standard Deviation	
Image Thresholding	0.87	0.51	[ms]
Quadrangle Detection	4.35	1.89	[ms]
Circle Detection	0.06	0.03	[ms]
Cross Extraction	1.81	1.01	[ms]
Perspective-n-Points	4.95	2.31	[ms]
Total	12.04	5.75	[ms]

F.4.3 Computational Load

As mentioned in Sec. F.3.3, our quadrotor is equipped with two onboard computers even though all algorithms composing our framework can be run off a single computer. The second computer is used solely for data recording and rapid prototyping. Our control and visual odometry pipelines have been demonstrated to run onboard the quadrotor in our previous work [43] and we refer the reader to that for further details. The trajectory planning algorithm we use in this work typically needs approximately 0.02 ms per trajectory. Since we replan our desired trajectory at 50 Hz, we can potentially compute up to 1000 candidate trajectories per replanning-cycle. Nevertheless, we fix the number of candidate trajectories to be computed at 20, which is usually sufficient to find a feasible trajectory during the platform following phase.

The statistics of the time required by our vision-based platform detection algorithm are reported in Table F.1. On average, it takes approximately 12 ms to detect the landing platform in each image, leading to a potential maximum rate of approximately 80 Hz. However, we found that a rate of 20 Hz is sufficient to obtain reliable and accurate results in tracking the landing platform.

F.4.4 Trajectory Planning

In this work, we use trajectories that minimize the jerk to provide our controller with reference states that drive the vehicle towards the accomplishment of the mission (cf. Sec. F.2.4). Previous work has shown that trajectories that minimize the snap, namely the fourth derivative of the position, lead to a smoother behavior for a quadrotor [111]. However, computing minimum snap trajectories typically requires longer then the closed form solution for minimum jerk trajectories we exploit. Also, to the best of our knowledge, no efficient feasibility verification method is available for minimum snap trajectories. In our experiments, we observed a better overall behavior of the entire pipeline when using minimum jerk trajectories. The reasons behind this are twofold: (i)

the very efficient computation of minimum jerk trajectories make it possible to re-plan the desired trajectory at high frequency to deal with changes in the motion of the moving target; (ii) the feasibility verification method lets us plan trajectories which satisfy the physical limits of the platform, i.e. avoid motors saturation.

F.4.5 Dealing with Missing Platform Detection

We deal with temporarily missing detections of the moving platform during the following and landing phases by using the Extended Kalman Filter described in Sec. F.2.3. Despite the lack of prior information about the motion of the platform and the constant velocity assumption, the dynamical model used for the prediction phase provides reliable results in both simulation and real world experiments. Therefore, our framework is capable of landing a quadrotor on a moving target even in the case when the platform is not temporarily visible.

F.5 Conclusions

In this work, we presented a quadrotor system capable of autonomously landing on a moving platform using only onboard sensing and computing. We relied on state-of-the-art computer vision algorithms, multi-sensor fusion for localization of the UAV, detection and motion estimation of the moving platform, and path planning for fully autonomous navigation. No external infrastructure, such as motion-capture systems or GPS, is needed. No prior information about the location of the moving landing target is required to execute the mission. We validated our framework in simulation as well as with real-world experiments using low-cost and lightweight consumer hardware. To the best of our knowledge, this is the first demonstration of a fully autonomous quadrotor system capable of landing on a moving target, using only onboard sensing and computing, without relying on any external infrastructure.

Bibliography

[1] A. Al-Tamimi and F. Lewis. "Discrete-time nonlinear HJB solution using approximate dynamic programming: convergence proof". In: *IEEE Transactions on Systems, Man, and Cybernetics*. June 2008. DOI: 10.1109/TSMCB.2008.926614.

[2] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. "Active vision". In: *International Journal of Computer Vision* 1.4 (Jan. 1988), pp. 333–356. ISSN: 1573-1405. DOI: 10.1007/BF00133571.

[3] H. Alvarez, L. M. Paz, and D. Cremers. "Collision Avoidance for Quadrotors with a Monocular Camera". In: *Int. Symp. Experimental Robotics (ISER)*. 2016, pp. 195–209. ISBN: 978-3-319-23778-7. DOI: 10.1007/978-3-319-23778-7_14.

[4] D. Arthur and S. Vassilvitskii. "How Slow is the K-means Method?" In: *Proceedings of the Twenty-second Annual Symposium on Computational Geometry*. SCG '06. ACM, 2006, pp. 144–153. ISBN: 1-59593-340-9. DOI: 10.1145/1137856.1137880.

[5] T. Avant, U. Lee, B. Katona, and K. Morgansen. "Dynamics, Hover Configurations, and Rotor Failure Restabilization of a Morphing Quadrotor". In: *2018 Annual American Control Conference (ACC)*. IEEE. 2018, pp. 4855–4862.

[6] Y. Bai and S. Gururajan. "Evaluation of a Baseline Controller for Autonomous "Figure-8" Flights of a Morphing Geometry Quadcopter: Flight Performance". In: *Drones* 3.3 (2019).

[7] S. Baker and I. Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework". In: *Int. J. Comput. Vis.* 56.3 (2004), pp. 221–255.

[8] M. Bangura, M. Melega, R. Naldi, and R. Mahony. "Aerodynamics of rotor blades for quadrotors". In: *arXiv e-prints* (Dec. 2016). URL: http://arxiv.org/abs/1601.00733.

[9] M. Bangura and R. Mahony. "Real-time Model Predictive Control for Quadrotors". In: *IFAC World Congress* (2014). DOI: 10.3182/20140824-6-za-1003.00203.

[10] A. J. Barry, P. R. Florence, and R. Tedrake. "High-speed autonomous obstacle avoidance with pushbroom stereo". In: *J. Field Robot.* 35.1 (Jan. 2018), pp. 52–68. ISSN: 1556-4967. DOI: 10.1002/rob.21741.

[11] S. Behnke, A. Egorova, A. Gloye, R. Rojas, and M. Simon. "Predicting Away Robot Control Latency". In: *RoboCup 2003: Robot Soccer World Cup VII*. Springer Berlin Heidelberg, 2004, pp. 712–719. ISBN: 978-3-540-25940-4.

[12] D. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I*. 2nd. Athena Scientific, 2005.

[13] H. Blum, A. Dietmüller, M. Milde, J. Conradt, G. Indiveri, and Y. Sandamirskaya. "A neuromorphic controller for a robotic vehicle equipped with a dynamic vision sensor". In: *Robotics: Science and Systems (RSS)*. 2017. DOI: 10.15607/rss.2017.xiii.035.

[14] F. J. Boria, R. J. Bachmann, P. G. Ifju, R. D. Quinn, R. Vaidyanathan, C. Perry, and J. Wagener. "A sensor platform capable of aerial and terrestrial locomotion". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. IEEE. 2005, pp. 3959–3964.

[15] A. Borowczyk, D. Nguyen, A. Phu-Van Nguyen, D. Q. Nguyen, D. Saussié, and J. Le Ny. "Autonomous Landing of a Multirotor Micro Air Vehicle on a High Velocity Ground Vehicle". In: *ArXiv* abs/1611.07329 (2016). URL: http://arxiv.org/abs/1611.07329.

[16] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck. "A 240x180 130dB 3us Latency Global Shutter Spatiotemporal Vision Sensor". In: *IEEE J. Solid-State Circuits* 49.10 (2014), pp. 2333–2341. ISSN: 0018-9200. DOI: 10.1109/JSSC.2014.2342715.

[17] D. Brescianini and R. D'Andrea. "Design, modeling and control of an omni-directional aerial vehicle". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2017. DOI: 10.1109/ICRA.2016.7487497.

[18] D. Brescianini, M. Hehn, and R. D'Andrea. "Quadrocopter Pole Acrobatics". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2013. DOI: 10.1109/IROS.2013.6696851.

[19] N. Bucki and M. W. Mueller. "Design and Control of a Passively Morphing Quadcopter". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2019, pp. 9116–9122. DOI: 10.1109/ICRA.2019.8794373.

[20] M. Burri, H. Oleynikova, M. W. Achtelik, and R. Siegwart. "Real-time visual-inertial mapping, re-localization and planning onboard MAVs in unknown environments". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2015, pp. 1872–1878. DOI: 10.1109/IROS.2015.7353622.

[21] A. Censi. "Efficient neuromorphic optomotor heading regulation". In: *American Control Conference (ACC)*. July 2015, pp. 3854–3861.

[22] A. Censi and D. Scaramuzza. "Low-Latency Event-Based Visual Odometry". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014. DOI: 10.1109/IROS.2016.7758089.

[23] L. Chittka, P. Skorupski, and N. E. Raine. "Speed–accuracy tradeoffs in animal decision making". In: *Trends in ecology & evolution* 24.7 (2009), pp. 400–407.

[24] T. Cieslewski, E. Kaufmann, and D. Scaramuzza. "Rapid exploration with multi-rotors: A frontier selection method for high speed flight". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2017, pp. 2135–2142. DOI: 10.1109/IROS.2017.8206030.

[25] X. Clady, C. Clercq, S.-H. Ieng, F. Houseini, M. Randazzo, L. Natale, C. Bartolozzi, and R. Benosman. "Asynchronous visual event-based time-to-contact". In: *Front. Neurosci.* 8.9 (2014). DOI: 10.3389/fnins.2014.00009.

[26] D. Comaniciu and P. Meer. "Mean shift: A robust approach toward feature space analysis". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 5 (2002), pp. 603–619.

[27] J. Conradt, R. Berner, M. Cook, and T. Delbruck. "An Embedded AER Dynamic Vision Sensor for Low-Latency Pole Balancing". In: *IEEE Workshop on Embedded Computer Vision (ECV)*. 2009.

[28] G. Costante, J. Delmerico, M. Werlberger, P. Valigi, and D. Scaramuzza. "Exploiting Photometric Information for Planning under Uncertainty". In: *Robotics Research: Volume 1* (2018), pp. 107–124. DOI: 10.1007/978-3-319-51532-8_7.

[29] M. Cutler and J. How. "Analysis and Control of a Variable-Pitch Quadrotor for Agile Flight". In: *ASME Journal of Dynamic Systems, Measurement and Control* 137.10 (Oct. 2015).

[30] L. Daler, S. Mintchev, C. Stefanini, and D. Floreano. "A bioinspired multi-modal flying and walking robot". In: *Bioinspiration & biomimetics* 10.1 (2015), p. 016005.

[31] J. M. Daly, Y. Ma, and S. L. Waslander. "Coordinated landing of a quadrotor on a skid-steered ground vehicle in the presence of time delays". In: *Auton. Robots* 38.2 (2015), pp. 179–191. ISSN: 1573-7527.

[32] T. Delbruck and P. Lichtsteiner. "Fast Sensory Motor Control Based on Event-Based Hybrid Neuromorphic-Procedural System". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2007, pp. 845–848. DOI: 10.1109/ISCAS.2007.378038.

[33] T. Delbruck and M. Lang. "Robotic Goalie with 3ms Reaction Time at 4% CPU Load Using Event-Based Dynamic Vision Sensor". In: *Front. Neurosci.* 7 (2013), p. 223. DOI: 10.3389/fnins.2013.00223.

[34] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, A. Ijspeert, D. Floreano, L. M. Gambardella, R. Siegwart, and D. Scaramuzza. "The current state and future outlook of rescue robotics". In: *J. Field Robot.* (2019). DOI: 10.1002/rob.21887.

[35] A Desbiez, F Expert, M Boyron, J Diperi, S Viollet, and F Ruffier. "X-Morf: A crash-separable quadrotor that morfs its X-geometry in flight". In: *2017 Workshop on Research, Education and Development of Unmanned Aerial Systems (RED-UAS)*. 2017, pp. 222–227. DOI: 10.1109/RED-UAS.2017.8101670.

[36] L. Doitsidis, S. Weiss, A. Renzaglia, M. W. Achtelik, E. Kosmatopoulos, R. Siegwart, and D. Scaramuzza. "Optimal surveillance coverage for teams of micro aerial vehicles in GPS-denied environments using onboard vision". In: *Auton. Robots* 33.1 (2012), pp. 173–188. ISSN: 1573-7527.

[37] D. Eberly. "Distance from a point to an ellipse, an ellipsoid, or a hyperellipsoid". In: *Geometric Tools, LLC* (2011).

[38] O. Esrafilian and H. D. Taghirad. "Autonomous flight and obstacle avoidance of a quadrotor by monocular SLAM". In: *International Conference on Robotics and Mechatronics (ICROM)*. Oct. 2016, pp. 240–245. DOI: 10.1109/ICRoM.2016.7886853.

[39] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. "A Density-based Algorithm for Discovering Clusters a Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. 1996, pp. 226–231.

[40] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." In: *Kdd*. Vol. 96. 1996, pp. 226–231.

[41] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza. "Automatic re-initialization and failure recovery for aggressive flight with a monocular vision-based quadrotor". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 1722–1729.

[42] M. Faessler, F. Fontana, C. Forster, and D. Scaramuzza. "Automatic Re-Initialization and Failure Recovery for Aggressive Flight with a Monocular Vision-Based Quadrotor". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 1722–1729. DOI: 10.1109/ICRA.2015.7139420.

[43] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza. "Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor MAV". In: *J. Field Robot.* 33.4 (2016), pp. 431–450. ISSN: 1556-4967. DOI: 10.1002/rob.21581.

[44] M. Faessler, A. Franchi, and D. Scaramuzza. "Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories". In: *IEEE Robot. Autom. Lett.* 3.2 (Apr. 2018), pp. 620–626. ISSN: 2377-3766. DOI: 10.1109/LRA.2017.2776353.

[45] M. Faessler, D. Falanga, and D. Scaramuzza. "Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight". In: *IEEE Robot. Autom. Lett.* 2.2 (Apr. 2017), pp. 476–482. ISSN: 2377-3766. DOI: 10.1109/LRA.2016.2640362.

[46] D. Falanga, E. Mueggler, M. Faessler, and D. Scaramuzza. "Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017. DOI: 10.1109/icra.2017.7989679.

[47] D. Falanga, S. Kim, and D. Scaramuzza. "How Fast is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid". In: *IEEE Robot. Autom. Lett.* 4.2 (Apr. 2019), pp. 1884–1891. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2898117.

[48] D. Falanga, K. Kleber, and D. Scaramuzza. "Low Latency Avoidance of Dynamic Obstacles for Quadrotors with Event Cameras". In: *AAAS Science Robotics, Under Review* (2019).

[49] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza. "PAMPC: Perception-Aware Model Predictive Control for Quadrotors". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Oct. 2018. DOI: 10.1109/IROS.2018.8593739.

[50] D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza. "The Foldable Drone: A Morphing Quadrotor that can Squeeze and Fly". In: *IEEE Robot. Autom. Lett.* 4.2 (Apr. 2019), pp. 209–216. ISSN: 2377-3766. DOI: 10.1109/LRA.2018.2885575.

[51] D. Falanga, A. Zanchettin, A. Simovic, J. Delmerico, and D. Scaramuzza. "Vision-based Autonomous Quadrotor Landing on a Moving Platform". In: *IEEE Int. Symp. Safety, Security, and Rescue Robot. (SSRR)*. Oct. 2017. DOI: 10.1109/SSRR.2017.8088164.

[52] D. Floreano and R. J. Wood. "Science, technology and the future of small autonomous drones". In: *Nature* 521 (2015), pp. 460–466. DOI: 10.1038/nature14542.

[53] P. Foehn, D. Falanga, N. Kuppuswamy, R. Tedrake, and D. Scaramuzza. "Fast Trajectory Optimization for Agile Quadrotor Maneuvers with a Cable-Suspended Payload". In: *Robotics: Science and Systems (RSS)*. June 2017. DOI: 10.15607/RSS.2017.XIII.030.

[54] P. Foehn and D. Scaramuzza. "Onboard state dependent LQR for agile quadrotors". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2018.

[55] E. W. Forgy. "Cluster analysis of multivariate data: efficiency versus interpretability of classifications". In: *biometrics* 21 (1965), pp. 768–769.

[56] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast Semi-Direct Monocular Visual Odometry". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 15–22. DOI: 10.1109/ICRA.2014.6906584.

[57] C. Forster, M. Pizzoli, and D. Scaramuzza. "Appearance-based Active, Monocular, Dense Depth Estimation for Micro Aerial Vehicles". In: *Robotics: Science and Systems (RSS)*. 2014. DOI: 10.15607/RSS.2014.X.029.

[58] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast semi-direct monocular visual odometry". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 15–22.

[59] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart. "Robot Operating System (ROS): The Complete Reference (Volume 1)". In: ed. by A. Koubaa. Springer International Publishing, 2016. Chap. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625. ISBN: 978-3-319-26054-9.

[60] A. Fusiello. *Elements of Geometric Computer Vision*. 16.09.2012. University of Edinburgh - School of Informatics. URL: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO4/tutorial.html#x1-130004.

[61] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. "Event-based Vision: A Survey". In: *arXiv e-prints* (2019). URL: http://arxiv.org/abs/1904.08405.

[62] D. Gallup, J.-M. Frahm, and M. Pollefeys. "Variable baseline/resolution stereo". In: *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2008. DOI: 10.1109/CVPR.2008.4587671.

[63] F. Galluppi, C. Denk, M. C. Meiner, T. C. Stewart, L. A. Plana, C. Eliasmith, S. Furber, and J. Conradt. "Event-based neural computing on an autonomous mobile platform". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. May 2014, pp. 2862–2867. DOI: 10.1109/ICRA.2014.6907270.

[64] K. A. Ghamry, Y. Dong, M. A. Kamel, and Y. Zhang. "Real-time autonomous take-off, tracking and landing of UAV on a moving UGV platform". In: *Control and Automation (MED), 2016 24th Mediterranean Conference on*. 2016, pp. 1236–1241.

[65] A. Glover and C. Bartolozzi. "Event-driven ball detection and gaze fixation in clutter". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2016, pp. 2203–2208. DOI: 10.1109/IROS.2016.7759345.

[66] A. Glover and C. Bartolozzi. "Robust visual tracking with a freely-moving event camera". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2017, pp. 3769–3776. DOI: 10.1109/IROS.2017.8206226.

[67] C. Greatwood, L. Bose, T. Richardson, W. Mayol, J. Chen, S. Carey, and P. Dudek. "Agile control of a UAV by tracking with a parallel visual processor". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2017.

[68] M. Guo, J. Huang, and S. Chen. "Live demonstration: A 768x215; 640 pixels 200Meps dynamic vision sensor". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. May 2017.

[69] A. Handa, R. Newcombe, A. Angeli, and A. Davison. "Real-Time Camera Tracking: When is High Frame-Rate Best?" In: *Eur. Conf. Comput. Vis. (ECCV)*. 2012. DOI: 10.1007/978-3-642-33786-4_17.

[70] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Second Edition. Cambridge University Press, 2003.

[71] M. A. Henson and D. E. Seborg, eds. *Nonlinear Process Control*. Prentice-Hall, Inc., 1997. ISBN: 0-13-625179-X.

[72] B. Herisse, F. Russotto, T. Hamel, and R. Mahony. "Hovering flight and vertical landing control of a VTOL unmanned aerial vehicle using optical flow". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2008, pp. 801–806.

[73] G. Hoffmann, D. G. Rajnarayan, S. L. Waslander, D. Dostal, J. S. Jang, and C. J. Tomlin. "The Stanford testbed of autonomous rotorcraft for multi agent control (STARMAC)". In: *The 23rd Digital Avionics Systems Conference (IEEE Cat. No. 04CH37576)*. Vol. 2. IEEE. 2004, 12–E.

[74] G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin. "Quadrotor helicopter flight dynamics and control: Theory and experiment". In: *AIAA Guidance, Navigation, and Control Conference*. Vol. 2. Aug. 2007, p. 4.

[75] B. Houska, H. Ferreau, and M. Diehl. "ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization". In: *Optimal Control Applications and Methods* 32.3 (2011).

[76] B. Houska, H. Ferreau, and M. Diehl. "An Auto-Generated Real-Time Iteration Algorithm for Nonlinear MPC in the Microsecond Range". In: *Automatica* 47.10 (2011). DOI: 10.1016/j.automatica.2011.08.020.

[77] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy. "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera". In: *Proc. Int. Symp. Robot. Research (ISRR)*. 2017, pp. 235–252. ISBN: 978-3-319-29363-9. DOI: 10.1007/978-3-319-29363-9_14.

[78] S. G. Johnson. "The NLopt nonlinear-optimization package". In: (). URL: http://ab-initio.mit.edu/nlopt.

[79] S. Jung, S. Cho, D. Lee, H. Lee, and D. H. Shim. "A direct visual servoing-based framework for the 2016 IROS Autonomous Drone Racing Challenge". In: *J. Field Robot.* 35.1 (May 2017), pp. 146–166. ISSN: 1556-4967. DOI: 10.1002/rob.21743.

[80] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Transactions of the ASME–Journal of Basic Engineering* 82.Series D (1960), pp. 35–45.

[81] M. Kamel, M. Burri, and R. Siegwart. "Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles". In: *arXiv* (2016). URL: http://arxiv.org/abs/1611.09240.

[82] M. Kamel, S. Verling, O. Elkhatib, C. Sprecher, P. Wulkop, Z. Taylor, R. Siegwart, and I. Gilitschenski. "The Voliro Omnidirectional Hexacopter: An Agile and Maneuverable Tiltable-Rotor Aerial Vehicle". In: *IEEE Robot. Autom. Mag.* (Oct. 2018). DOI: 10.1109/MRA.2018.2866758.

[83] S. Karaman and E. Frazzoli. "High-speed flight in an ergodic forest". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2012. DOI: 10.1109/ICRA.2012.6225235.

[84] E. Kaufmann, A. Loquercio, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. "Deep Drone Racing:Learning Agile Flight in Dynamic Environments". In: *arXiv e-prints* (2018). URL: http://arxiv.org/abs/1806.08548.

[85] O. Khatib. "Real-time obstacle avoidance for manipulators and mobile robots". In: *Autonomous Robot Vehicles*. Springer, 1986, pp. 396–404.

[86] P. Khosla and R. Volpe. "Superquadric artificial potentials for obstacle avoidance and approach". In: *icra*. 1988, pp. 1778–1784. DOI: 10.1109/ROBOT.1988.12323.

[87] J. Kim, Y. Jung, D. Lee, and D. H. Shim. "Landing Control on a Mobile Platform for Multi-copters using an Omnidirectional Image Sensor". In: *J. Intell. Robot. Syst.* (2016), pp. 1–13. ISSN: 1573-0409.

[88] S. Kim, D. Falanga, and D. Scaramuzza. "Computing the Forward Reachable Set for a Multirotor Under First-Order Aerodynamic Effects". In: *IEEE Robot. Autom. Lett.* 3.4 (Oct. 2018), pp. 2934–2941. DOI: 10.1109/LRA.2018.2848302.

[89] D. Kraft. *A Software Package for Sequential Quadratic Programming*. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988.

[90] V. Kumar and N. Michael. "Opportunities and challenges with autonomous micro aerial vehicles". In: *J. Field Robot.* 31.11 (Sept. 2012).

[91] S. Lange, N. Sünderhauf, and P. Protzel. "Autonomous landing for a multirotor UAV using vision". In: *Int. Conf. on Simulation, Modeling, and Programming for Auton. Robots (SIMPAR)*. 2008, pp. 482–491.

[92] D. Lee, T. Ryan, and H. J. Kim. "Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2012, pp. 971–976.

[93] S. Lee, D. K. Giri, and H. Son. "Modeling and control of quadrotor UAV subject to variations in center of gravity and mass". In: *Int. Conf. on Ubiquitous Robots and Ambient Intelligence (URAI)*. 2017.

[94] D. Lentink, U. K. Mueller, E. J. Stamhuis, R. D. Kat, W. V. Gestel, L. L. M. Veldhuis, P. Henningsson, A. Hedenstroem, J. J. Videler, and J. L. V. Leeuwen. "How swifts control their glide performance with morphing wings". In: *Nature* 446 (2007), pp. 1082–1085. DOI: 10.1038/nature05733.

[95] P. Lichtsteiner, C. Posch, and T. Delbruck. "A 128x128 120dB 30mW asynchronous vision sensor that responds to relative intensity change". In: *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*. 2006, pp. 2060–2069. DOI: 10.1109/ISSCC.2006.1696265.

[96] P. Lichtsteiner, C. Posch, and T. Delbruck. "A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor". In: *IEEE J. Solid-State Circuits* 43.2 (2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[97] Y. Lin, F. Gao, T. Qin, W. Gao, T. Liu, W. Wu, Z. Yang, and S. Shen. "Autonomous aerial navigation using monocular visual-inertial fusion". In: *J. Field Robot.* 35.1 (2018), pp. 23–51. DOI: 10.1002/rob.21732.

[98] S. Liu, M. Watterson, S. Tang, and V. Kumar. "High speed navigation for quadrotors with limited onboard sensing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2016, pp. 1484–1491. DOI: 10.1109/ICRA.2016.7487284.

[99] G. Loianno, C. Brunner, G. McGrath, and V. Kumar. "Estimation, Control, and Planning for Aggressive Flight With a Small Quadrotor With a Single Camera and IMU". In: *IEEE Robot. Autom. Lett.* 2.2 (Apr. 2017), pp. 404–411. DOI: 10.1109/lra.2016.2633290.

[100] B. T. Lopez and J. P. How. "Aggressive 3-D collision avoidance for high-speed navigation". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2017. DOI: 10.1109/icra.2017.7989677.

[101] R. Lozano, J. Guerrero, and N. Chopra. "Quadrotor Flight Formation Control Via Positive Realness Multivehicle Systems". In: (2012). DOI: 10.3182/20121003-3-SF-4024.00004.

[102] B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Int. Joint Conf. Artificial Intell. (IJCAI)*. 1981, pp. 674–679.

[103] S. Lynen, M. Achtelik, S. Weiss, M. Chli, and R. Siegwart. "A Robust and Modular Multi-Sensor Fusion Approach Applied to MAV Navigation". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Nov. 2013.

[104] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart. "A robust and modular multi-sensor fusion approach applied to mav navigation". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2013, pp. 3923–3929.

[105] E. Lyu, Y. Lin, W. Liu, and M. Q. H. Meng. "Vision based autonomous gap-flying-through using the micro unmanned aerial vehicle". In: *Electrical and Computer Engineering, IEEE Canadian Conference on*. May 2015, pp. 744–749.

[106] R. Mahony, V. Kumar, and P. Corke. "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor". In: *IEEE Robot. Autom. Mag.* 19.3 (2012), pp. 20–32. ISSN: 1070-9932. DOI: 10.1109/MRA.2012.2206474.

[107] J. E. Mebius. "Derivation of the Euler-Rodrigues formula for three-dimensional rotations from the general formula for four-dimensional rotations". In: *arXiv e-prints* (2007). URL: http://arxiv.org/abs/math/0701759.

[108] D. Mellinger, M. Shomin, and V. Michael N. Kumar. "Cooperative Grasping and Transport Using Multiple Quadrotors". In: *Distributed Autonomous Robotic Systems: The 10th International Symposium*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 545–558. ISBN: 978-3-642-32723-0.

[109] D. Mellinger, M. Shomin, and V. Kumar. "Control of quadrotors for robust perching and landing". In: (2010), pp. 205–225.

[110] D. Mellinger and V. Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2011, pp. 2520–2525. DOI: 10.1109/ICRA.2011.5980409.

[111] D. Mellinger and V. Kumar. "Minimum snap trajectory generation and control for quadrotors". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2011, pp. 2520–2525.

[112] D. Mellinger, N. Michael, and V. Kumar. "Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors". In: *Int. Symp. Experimental Robotics (ISER)*. Dec. 2010.

[113] M. B. Milde, O. J. N. Bertrand, H. Ramachandran, M. Egelhaaf, and E. Chicca. "Spiking Elementary Motion Detector in Neuromorphic Systems". In: *Neural Computation* 30.9 (Sept. 2018), pp. 2384–2417. DOI: 10.1162/neco_a_01112.

[114] S Mintchev, S de Rivaz, and D Floreano. "Insect-Inspired Mechanical Resilience for Multicopters". In: *IEEE Robotics and Automation Letters* 2.3 (2017), pp. 1248–1255. DOI: 10.1109/LRA.2017.2658946.

[115] S. Mintchev and D. Floreano. "Adaptive Morphology: A Design Principle for Multimodal and Multifunctional Robots". In: *IEEE Robot. Autom. Mag.* 23 (2016), pp. 42–54. DOI: 10.1109/MRA.2016.2580593.

[116] S. Mintchev, J. Shintake, and D. Floreano. "Bioinspired dual-stiffness origami". In: *Science Robotics* 3.20 (2018), eaau0275.

[117] S. Mintchev, J. Shintake, and D. Floreano. "Bioinspired dual-stiffness origami". In: *Science Robotics* 3.20 (July 2018). URL: http://robotics.sciencemag.org/content/3/20/eaau0275.abstract.

[118] S. Mintchev, S. de Rivaz, and D. Floreano. "Insect-inspired mechanical resilience for multicopters". In: *IEEE Robotics and automation letters* 2.3 (2017), pp. 1248–1255.

[119] A. Mitrokhin, C. Fermuller, C. Parameshwara, and Y. Aloimonos. "Event-based Moving Object Detection and Tracking". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2018.

[120] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar. "Fast, autonomous flight in GPS-denied and cluttered environments". In: *J. Field Robot.* 35.1 (Apr. 2017), pp. 101–120. ISSN: 1556-4967. DOI: 10.1002/rob.21774.

[121] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar. "Fast, autonomous flight in GPS-denied and cluttered environments". In: *J. Field Robot.* 35.1 (2018), pp. 101–120. DOI: 10.1002/rob.21774.

[122] H. Moon, J. Martinez-Carranza, T. Cieslewski, M. Faessler, D. Falanga, A. Simovic, D. Scaramuzza, S. Li, M. Ozo, C. De Wagter, G. de Croon, S. Hwang, S. Jung, H. Shim, H. Kim, M. Park, T.-C. Au, and S. J. Kim. "Challenges and implemented technologies used in autonomous drone racing". In: *Intelligent Service Robotics* 12.2 (Apr. 2019), pp. 137–148. ISSN: 1861-2784. DOI: 10.1007/s11370-018-00271-6.

[123] N. Moshtagh et al. "Minimum volume enclosing ellipsoid". In: *Convex optimization* 111 (2005), p. 112.

[124] A. I. Mourikis and S. I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. Apr. 2007, pp. 3565–3572.

[125] E. Mueggler, C. Forster, N. Baumli, G. Gallego, and D. Scaramuzza. "Lifetime Estimation of Events from Dynamic Vision Sensors". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 4874–4881. DOI: 10.1109/ICRA.2015.7139876.

[126] E. Mueggler, N. Baumli, F. Fontana, and D. Scaramuzza. "Towards Evasive Maneuvers with Quadrotors using Dynamic Vision Sensors". In: *Eur. Conf. Mobile Robots (ECMR)*. 2015, pp. 1–8. DOI: 10.1109/ECMR.2015.7324048.

[127] E. Mueller, A. Censi, and E. Frazzoli. "Low-latency Heading Feedback Control with Neuromorphic Vision Sensors using Efficient Approximated Incremental Inference". In: *IEEE Conf. Decision Control (CDC)*. 2015. DOI: 10.1109/CDC.2015.7402002.

[128] M. Mueller, S. Lupashin, and R. D'Andrea. "Quadrocopter ball juggling". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2011, pp. 4972–4978. DOI: 10.1109/IROS.2012.6385963.

[129] M. W. Mueller, M. Hehn, and R. D'Andrea. "A computationally efficient motion primitive for quadrocopter trajectory generation". In: *IEEE Trans. Robot.* 31.6 (2015), pp. 1294–1310.

[130] M. W. Mueller, M. Hehn, and R. D'Andrea. "A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation". In: *IEEE Trans. Robot.* 31.6 (2015), pp. 1294–1310.

[131] V. Murali, I. Spasojevic, W. Guerra, and S. Karaman. "Perception-aware trajectory generation for aggressive quadrotor flight using differential flatness". In: *American Control Conference (ACC)*. July 2019, pp. 3936–3943.

[132] T. Muskardin, G. Balmer, S. Wlach, K. Kondak, M. Laiacker, and A. Ollero. "Landing of a fixed-wing UAV on a mobile ground vehicle". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2016, pp. 1237–1242.

[133] T. Nägeli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges. "Real-time Motion Planning for Aerial Videography with Dynamic Obstacle Avoidance and Viewpoint Optimization". In: *IEEE Robot. Autom. Lett.* 2.3 (2017). DOI: 10.1109/LRA.2017.2665693.

[134] T. Nägeli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges. "Real-time Planning for Automated Multi-View Drone Cinematography". In: *SIGGRAPH*. 2017.

[135] G. Nandakumar, A. Srinivasan, and A. Thondiyath. "Theoretical and experimental investigations on the effect of overlap and offset on the design of a novel quadrotor configuration, VOOPS". In: *Journal of Intelligent & Robotic Systems* (2017). DOI: 10.1007/s10846-017-0707-2.

[136] M. Neunert, C. de Crousaz, F. Furrer, M. Kamel, F. Farshidian, R. Siegwart, and J. Buchli. "Fast nonlinear Model Predictive Control for unified trajectory optimization and tracking". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2016. DOI: 10.1109/icra.2016.7487274.

[137] B. Nisar, P. Foehn, D. Falanga, and D. Scaramuzza. ""VIMO: Simultaneous Visual Inertial Model-based Odometry and Force Estimation". In: *IEEE Robot. Autom. Lett.* 4.3 (July 2019), pp. 2785–2792. DOI: 10.1109/LRA.2019.2918689.

[138] H. Oleynikova, D. Honegger, and M. Pollefeys. "Reactive Avoidance Using Embedded Stereo Vision for MAV Flight". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 50–56.

[139] E. Olson. "AprilTag: A Robust and Flexible Visual Fiducial System". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. May 2011.

[140] D. Palossi, A. Loquercio, F. Conti, F. Conti, E. Flamand, E. Flamand, D. Scaramuzza, L. Benini, and L. Benini. "A 64mW DNN-based Visual Navigation Engine for Autonomous Nano-Drones". In: *IEEE Internet of Things Journal* (2019), pp. 1–1. ISSN: 2327-4662. DOI: 10.1109/JIOT.2019.2917066.

[141] B. Penin, R. Spica, P. Robuffo Giordano, and F. Chaumette. "Vision-Based Minimum-Time Trajectory Generation for a Quadrotor UAV". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2017.

[142] B. Penin, P. R. Giordano, and F. Chaumette. "Vision-Based Reactive Planning for Aggressive Target Tracking While Avoiding Collisions and Occlusions". In: *IEEE Robot. Autom. Lett.* 3.4 (Oct. 2018), pp. 3725–3732. DOI: 10.1109/LRA.2018.2856526.

[143] C. J. Pennycuick. "A wind-tunnel study of gliding flight in the pigeon Columba livia". In: *Journal of Exp. Biology* 49 (1968), pp. 509–526.

[144] M. Pizzoli, C. Forster, and D. Scaramuzza. "REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 2609–2616. DOI: 10.1109/ICRA.2014.6907233.

[145] C. Potena, D. Nardi, and A. Pretto. "Effective Target Aware Visual Navigation for UAVs". In: *Eur. Conf. Mobile Robots (ECMR)*. 2017.

[146] C. Potena, D. Nardi, and A. Pretto. "Joint Vision-Based Navigation, Control and Obstacle Avoidance for UAVs in Dynamic Environments". In: *CoRR* abs/1905.01187 (2019). arXiv: 1905.01187. URL: http://arxiv.org/abs/1905.01187.

[147] P. Pounds, R. Mahony, and P. Corke. "Modelling and Control of a Quad-Rotor Robot". In: *Australasian Conf. Robot. Autom.* 2006.

[148] D. Pucci, S. Traversaro, and F. Nori. "Momentum Control of an Underactuated Flying Humanoid Robot". In: *IEEE Robot. Autom. Lett.* 3.1 (Jan. 2018). DOI: 10.1109/LRA.2017.2734245.

[149] D. Reynolds. "Gaussian mixture models". In: *Encyclopedia of biometrics* (2015), pp. 827–832.

[150] T. S. Richardson, C. G. Jones, A. Likhoded, E. Sparks, A. Jordan, I. Cowling, and S. Willcox. "Automated Vision-based Recovery of a Rotary Wing Unmanned Aerial Vehicle onto a Moving Platform". In: *J. Field Robot.* 30.5 (2013), pp. 667–684. ISSN: 1556-4967.

[151] C. Richter, W. Vega-Brown, and N. Roy. "Bayesian Learning for Safe High-Speed Navigation in Unknown Environments". In: *International Symposium of Robotics Research, ISRR*. 2015, pp. 325–341. DOI: 10.1007/978-3-319-60916-4_19.

[152] C. Richter and N. Roy. "Safe Visual Navigation via Deep Learning and Novelty Detection". In: *Robotics: Science and Systems (RSS)*. July 2017. DOI: 10.15607/RSS.2017.XIII.064.

[153] V. Riviere, A. Manecy, and S. Viollet. "Agile Robotic Fliers: A Morphing-Based Approach". In: *Soft Robotics* (2018). DOI: 10.1089/soro.2017.0120.

[154] A. Rosinol Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: *IEEE Robot. Autom. Lett.* 3.2 (Apr. 2018), pp. 994–1001. DOI: 10.1109/LRA.2018.2793357.

[155] T. Rosinol Vidal, H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: *IEEE Robot. Autom. Lett.* PP.99 (2018), pp. 1–1. DOI: 10.1109/LRA.2018.2793357.

[156] B. Rueckauer and T. Delbruck. "Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor". In: *Frontiers in neuroscience* 10 (2016), p. 176.

[157] M Ryll, H. H. Bülthoff, and P. R. Giordano. "A Novel Overactuated Quadrotor Unmanned Aerial Vehicle: Modeling, Control, and Experimental Validation". In: *IEEE Transactions on Control Systems Technology* 23.2 (2015), pp. 540–556. ISSN: 1063-6536 VO - 23. DOI: 10.1109/TCST.2014.2330999.

[158] M Ryll, H. H. Bülthoff, and P. R. Giordano. "Modeling and control of a quadrotor UAV with tilting propellers". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2012, pp. 4606–4613. DOI: 10.1109/ICRA.2012.6225129.

[159] A. Sakaguchi, T. Takimoto, and T. Ushio. "A Novel Quadcopter with A Tilting Frame using Parallel Link Mechanism". In: *2019 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2019, pp. 674–683.

[160] P. Salaris, M. Cognetti, R. Spica, and P. R. Giordano. "Online Optimal Perception-Aware Trajectory Generation". In: *IEEE Trans. Robot.* (2019), pp. 1–16. DOI: 10.1109/TRO.2019.2931137.

[161] L. Salt and D. Howard. "Self-Adaptive Differential Evolution for Bio-Inspired Neuromorphic Collision Avoidance". In: *CoRR* abs/1704.04853 (2017). URL: http://arxiv.org/abs/1704.04853.

[162] N. J. Sanket, C. M. Parameshwara, C. D. Singh, A. V. Kuruttukulam, C. Fermuller, D. Scaramuzza, and Y. Aloimonos. "EVDodge: Embodied AI For High-Speed Dodging On A Quadrotor Using Event Cameras". In: *arXiv e-prints* (2019). URL: http://arxiv.org/abs/1906.02919.

[163] N. J. Sanket, C. D. Singh, K. Ganguly, C. Fermueller, and Y. Aloimonos. "GapFlyt: Active Vision Based Minimalist Structure-less Gap Detection For Quadrotor Flight". In: *IEEE Robot. Autom. Lett.* 3.4 (June 2018). DOI: 10.1109/LRA.2018.2843445.

[164] S. Saripalli, J. F. Montgomery, and G. S. Sukhatme. "Vision-based autonomous landing of an unmanned aerial vehicle". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. Vol. 3. 2002, pp. 2799–2804.

[165] T. Sayre-McCord, W. Guerra, A. Antonini, J. Arneberg, A. Brown, G. Cavalheiro, Y. Fang, A. Gorodetsky, D. McCoy, S. Quilter, F. Riether, E. Tal, Y. Terzioglu, L. Carlone, and S. Karaman. "Visual-inertial navigation algorithm development using photorealistic camera simulation in the loop". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018.

[166] P. Sermanet, R. Hadsell, J. Ben, A. Erkan, B. Flepp, U. Muller, and Y. LeCun. "Speed-range dilemmas for vision-based navigation in unstructured terrain". In: *6th IFAC Symposium on Intelligent Autonomous Vehicles*. Vol. 6. 2007, pp. 300–305. ISBN: 9783902661654.

[167] P. Serra, R. Cunha, T. Hamel, D. Cabecinhas, and C. Silvestre. "Landing of a Quadrotor on a Moving Target Using Dynamic Image-Based Visual Servo Control". In: *IEEE Trans. Robot.* 32.6 (Dec. 2016), pp. 1524–1535. ISSN: 1552-3098.

[168] C. S. Sharp, O. Shakernia, and S. Sastry. "A vision system for landing an unmanned aerial vehicle". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. Vol. 2. 2001, pp. 1720–1727.

[169] M. Sheckells, G. Garimella, and M. Kobilarov. "Optimal Visual Servoing for differentially flat underactuated systems". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2016.

[170] J. Shu and P. Chirarattananon. "A Quadrotor with an Origami-Inspired Protective Mechanism". In: *CoRR* abs/1907.07056 (2019). arXiv: 1907.07056. URL: http://arxiv.org/abs/1907.07056.

[171] B. Siciliano, L. Sciavicco, and L. Villani. *Robotics: modelling, planning and control*. Advanced Textbooks in Control and Signal Processing. London: Springer, 2009. ISBN: 1-8462-8641-7.

[172] B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. 2nd. Springer Publishing Company, Incorporated, 2016. ISBN: 3319325507, 9783319325507.

[173] H. Sorenson. *Kalman Filtering: Theory and Application*. IEEE Press selected reprint series. IEEE Press, 1985. ISBN: 9780879421915.

[174] R. Spica, P. Robuffo Giordano, and F. Chaumette. "Coupling Active Depth Estimation and Visual Servoing via a Large Projection Operator". In: *Int. J. Robot. Research* 36.11 (2017).

[175] R. Spica, D. Falanga, E. Cristofalo, E. Montijano, D. Scaramuzza, and M. Schwager. "A Game Theoretic Approach to Autonomous Two-Player Drone Racing". In: *Robotics: Science and Systems (RSS)*. June 2018. DOI: 10.15607/RSS.2018.XIV.040.

[176] M. W. Spong. "Partial feedback linearization of underactuated mechanical systems". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. Vol. 1. Sept. 1994, 314–321 vol.1. DOI: 10.1109/IROS.1994.407375.

[177] M. Stefano and F. Dario. "Adaptive morphology: A design principle for multimodal and multifunctional robots". In: *IEEE Robot. Autom. Mag.* 23.3 (2016), pp. 42–54.

[178] T. Stoffregen, G. Gallego, T. Drummond, L. Kleeman, and D. Scaramuzza. "Event-Based Motion Segmentation by Motion Compensation". In: *IEEE International Conference on Computer Vision (ICCV)*. 2019.

[179] K. Su and S. Shen. "Catching a Flying Ball with a Vision-Based Quadrotor". In: *Int. Symp. Experimental Robotics (ISER)*. 2016.

[180] S. Suzuki and K. Abe. "Topological structural analysis of digitized binary images by border following". In: *Computer Vision, Graphics, and Image Processing* 30.1 (1985), pp. 32–46. ISSN: 0734-189X. DOI: 10.1016/0734-189X(85)90016-7.

[181] R. Szeliski. *Computer Vision: Algorithms and Applications*. 2010. ISBN: 978-1-84882-935-0.

[182] R. Tallamraju, E. Price, R. Ludwig, K. Karlapalem, H. H. Buelthoff, M. Black, and A. Ahmad. "Active Perception based Formation Control for Multiple Aerial Vehicles". In: *IEEE Robot. Autom. Lett.* (2019). DOI: 10.1109/LRA.2019.2932570.

[183] D. Tang, F. Li, N. Shen, and S. Guo. "UAV attitude and position estimation for vision-based landing". In: *Electronic and Mechanical Engineering and Information Technology (EMEIT), International Conference on*. Vol. 9. 2011, pp. 4446–4450.

[184] S. Tang and V. Kumar. "Mixed integer Quadratic Program Trajectory Generation for a Quadrotor With a Cable-Suspended Payload". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 2216–2222. DOI: 10.1109/ICRA.2015.7139492.

[185] A. M. Tonello and B. Salamat. "A Swash Mass Unmanned Aerial Vehicle: Design, Modeling and Control". In: *arXiv preprint arXiv:1909.06154* (2019).

[186] L. W. Traub. "Calculation of constant power lithium battery discharge curves". In: *Batteries* 2.2 (2016). DOI: 10.3390/batteries2020017.

[187] M. Vincze. "Dynamics and system performance of visual servoing". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. Vol. 1. 2000, 644–649 vol.1. DOI: 10.1109/ROBOT.2000.844125.

[188] P. Vlantis, P. Marantos, C. P. Bechlioulis, and K. J. Kyriakopoulos. "Quadrotor landing on an inclined platform of a moving ground vehicle". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 2202–2207.

[189] D. Wallace. "Dynamics and Control of a Quadrotor with Active Geometric Morphing". MA thesis. University of Washington, 2016.

[190] M. Watterson, S. Liu, K. Sun, T. Smith, and V. Kumar. "Trajectory Optimization On Manifolds with Applications to SO(3) and R3XS2". In: *Robotics: Science and Systems (RSS)*. Pittsburgh, Pennsylvania, June 2018. DOI: 10.15607/RSS.2018.XIV.023.

[191] K. E. Wenzel, A. Masselli, and A. Zell. "Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle". In: *J. Intell. Robot. Syst.* 61.1-4 (2011), pp. 221–238.

[192] C. D. Williams and A. A. Biewener. "Pigeons trade efficiency for stability in response to level of challenge during confined flight". In: *Proc. Natl. Acad. Sci. U.S.A.* 2015. DOI: 10.1073/pnas.1407298112.

[193] H. Xiong, J. Hu, and X. Diao. "Optimize Energy Efficiency of Quadrotors Via Arm Rotation". In: *Journal of Dynamic Systems, Measurement, and Control* 141.9 (2019), p. 091002. DOI: 10.1115/1.4043227.

[194] Z. Zhang and D. Scaramuzza. "Perception-aware receding horizon navigation for mavs". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. IEEE. 2018, pp. 2534–2541.

[195] M. Zhao, K. Kawasaki, X. Chen, S. Noda, K. Okada, and M. Inaba. "Whole-body aerial manipulation by transformable multirotor with two-dimensional multi-links". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. June 2017. DOI: 10.1109/ICRA.2017.7989606.

[196] M. Zhao, T. Anzai, F. Shi, X. Chen, K. Okada, and M. Inaba. "Design, Modeling, and Control of an Aerial Robot DRAGON: A Dual-Rotor-Embedded Multilink Robot With the Ability of Multi-Degree-of-Freedom Aerial Transformation". In: *IEEE Robot. Autom. Lett.* 3.2 (Apr. 2018). DOI: 10.1109/LRA.2018.2793344.

[197] N. Zhao, Y. Luo, H. Deng, and Y. Shen. "The deformable quad-rotor: Design, kinematics and dynamics characterization, and flight performance validation". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2017. DOI: 10.1109/IROS.2017.8206052.

[198] A. Zhu, N. Atanasov, and K. Daniilidis. "Event-based Visual Inertial Odometry". In: *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2017. DOI: 10.1109/CVPR.2017.616.

Universität Zürich[UZH]

Department of Informatics

Curriculum vitae

Personal details

Davide Falanga
Date of birth: 01 November 1989

Education

April 15 – February 20:	Doctoral program at the University of Zurich, Department of Informatics, Robotics and Perception Group
September 12 – March 15:	Master of Science, University of Naples “Federico II”, Automation Engineering
September 08 – September 12:	Bachelor of Science, University of Naples “Federico II”, Automation Engineering

Professional experience

October 14 – March 15:	Robotics Intern at Swisslog AG, Buchs AG, Switzerland
November 07 – March 14:	Freelance Web-Journalist at GruppoHTML.it

Awards

2019 NASA Tech Briefs Create the Future award for the category “Aerospace & Defense”.

2019 Drone Hero Award, “Most Innovative Drone” category.

2018 IROS Autonomous Drone Racing, 1st place.

Universität Zürich[UZH]

Department of Informatics

Bibliographic citation

The single papers constituting the dissertation are to be cited as follows.

Journal Publications

1. Davide Falanga, Kevin Kleber, and Davide Scaramuzza. “Low Latency Avoidance of Dynamic Obstacles for Quadrotors with Event Cameras”. This paper has been submitted to the AAAS Science Robotics.

2. Davide Falanga, Suseong Kim, and Davide Scaramuzza. “How Fast is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid”. In: IEEE Robotics and Automation Letters (RA-L) 4.2 (2019), pp. 1884–1891. DOI: 10.1109/LRA.2019.2898117.

3. Davide Falanga, Kevin Kleber, Stefano Mintchev, Dario Floreano, and Davide Scaramuzza, “The Foldable Drone: A Morphing Quadrotor that can Squeeze and Fly”. In: IEEE Robotics and Automation Letters (RA-L)4.2(2019), pp. 209–216. DOI:10.1109/LRA.2018.2885575.

4. Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, Davide Falanga, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, Guido de Croon, Sunyou Hwang, Sunggoo Jung, Hyunchul Shim, Haeryang Kim, Minhyuk Park, Tsz-Chiu Au, and Si Jung Kim. “Challenges and implemented technologies used in autonomous drone racing”. In: Springer: Intelligent Service Robotics Series 12.2 (2019), pp. 137–148. DOI: 10.1007/s11370-018-00271-6.

5. Barza Nisar, Philipp Foehn, Davide Falanga, and Davide Scaramuzza. “VIMO: Simultaneous Visual Inertial Model-based Odometry and Force Estimation”. In: IEEE Robotics and Automation Letters (RA-L) 4.3 (2019), pp. 2785–2792. DOI: 10.1109/LRA.2019.2918689.

6. Suseong Kim, Davide Falanga,Davide Scaramuzza. “Computing The Forward Reachable Set for a Multirotor Under First-Order Aerodynamic Effects”. In: IEEE Robotics and Automation Letters (RA-L) 3.4 (2018), 2934–2941. DOI: 10.1109/LRA.2018.2848302.

7. Matthias Faessler, Davide Falanga, and Davide Scaramuzza. “Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight”. In: IEEE Robotics and Automation Letters (RA-L) 2.2 (2017), pp. 476–482. DOI: 10.1109/LRA.2016.2640362.

Peer-Reviewed Conference Papers

1. R. Spica, Davide Falanga, Eric Cristofalo, Eduardo Montijano, Davide Scaramuzza, and Mac Schwager. “A Game Theoretic Approach to Autonomous Two-Player Drone Racing”. In: Robotics: Science and Systems (RSS) 2018. DOI: 10.15607/RSS.2018.XIV.040.

2. Davide Falanga, Philipp Foehn, Peng Lu, and Davide Scaramuzza. “PAMPC: Perception-Aware Model Predictive Control for Quadrotors”. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2018. DOI: 10.1109/IROS.2018.8593739.

3. Davide Falanga, Elias Mueggler, Matthias Faessler, and Davide Scaramuzza. “Aggressive Quadrotor Flight through Narrow Gaps with Onboard Sensing and Computing”. In: IEEE International Conference on Robotics and Automation (ICRA) 2017. DOI: 10.1109/ICRA.2017.7989679.

4. Philipp Foehn, Davide Falanga, Naveen Kuppuswamy, Russ Tedrake, and Davide Scaramuzza. “Fast Trajectory Optimization for Agile Quadrotor Maneuvers with a Cable Suspended Payload”. In: Robotics: Science and Systems(RSS)2017. DOI:10.15607/RSS.2017.XIII.030.

5. Davide Falanga, Alessio Zanchettin, Alessandro Simovic, Jeffrey Delmerico, and Davide Scaramuzza. “Vision-based Autonomous Quadrotor Landing on a Moving Platform”. In: IEEE/RSJ International Symposium. on Safety, Security and Rescue Robotics (SSRR) 2017. DOI: 10.1109/SSRR.2017.8088164.